UC Berkeley

UC Berkeley Electronic Theses and Dissertations

Title

Deep Reinforcement Learning for Autonomous Vehicle Traffic Control and Stabilization: From Simulation to a 100-Vehicle Highway Field Deployment

Permalink

https://escholarship.org/uc/item/57t2048n

ISBN

9798293893287

Author

Lichtle, Nathan Christian

Publication Date

2025-08-01

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

Deep Reinforcement Learning for Autonomous Vehicle Traffic Control and Stabilization: From Simulation to a 100-Vehicle Highway Field Deployment

By

Nathan Lichtlé

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Alexandre Bayen, Co-Chair
Professor Amaury Hayat, Co-Chair
Professor Maria Laura Delle Monache
Professor Sergey Levine

Summer 2025

Deep Reinforcement Learning for Autonomous Vehicle Traffic Control and Stabilization:
From Simulation to a 100-Vehicle Highway Field Deployment

Copyright 2025
by
Nathan Lichtlé

Abstract

Deep Reinforcement Learning for Autonomous Vehicle Traffic Control and Stabilization: From Simulation to a 100-Vehicle Highway Field Deployment

by

Nathan Lichtlé

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Alexandre Bayen, Co-Chair

Professor Amaury Hayat, Co-Chair

Highway traffic instabilities, commonly manifested as stop-and-go waves or capacity drop at bottlenecks, represent a major source of energy waste and congestion in transportation systems. Even small fractions of autonomous vehicles (AVs) have the potential to dampen these phenomena and improve overall traffic flow. However, developing control methods that can transition from simulation to deployment in mixed-autonomy environments remains a fundamental challenge. The core problem lies in creating robust control algorithms that can handle the scale and complexity of multi-agent traffic interactions while maintaining safety and performance when deployed on actual roads with human drivers.

This work establishes a complete research pipeline from simulation-based algorithm development to large-scale field validation of deep reinforcement learning (RL) methods for AV traffic control. The approach begins with developing multi-agent RL techniques for decentralized bottleneck control. It introduces Nocturne, a high-throughput driving simulator built on trajectory data to enable scalable multi-agent learning. Wave-smoothing cruise controllers are then trained directly on highway trajectory data, leading to a validated deployment pipeline that transfers learned policies to production vehicles without retuning. The methodology culminates in a large-scale field experiment where RL controllers developed in this work operate 100 connected and automated vehicles deployed in live rush hour traffic on the I-24 highway in Tennessee, achieving measurable traffic smoothing through distributed control. The work concludes by introducing neural network-based methods for traffic flow prediction via partial differential equations. Together, these contributions demonstrate both the feasibility and effectiveness of learning-based traffic control in mixed-autonomy environments, offering a clear path toward substantial improvements in highway energy efficiency and traffic congestion reduction.

Contents

Contents **i**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Thesis Organization . . . 3
1.3 Acknowledgement of Contributions . . . 4

2 Background **5**
2.1 Traffic Flow Theory . . . 5
2.2 Car-Following Dynamics . . . 7
2.3 String Stability and Wave Formation . . . 8
2.4 Mixed Autonomy Traffic Control . . . 10
2.5 Reinforcement Learning . . . 11
2.6 Policy Gradient Methods . . . 14
2.7 Multi-Agent Learning Challenges . . . 16

I Deep Reinforcement Learning for Traffic Control **18**

3 Decentralized AVs for Bottleneck Control **19**
3.1 Introduction . . . 19
3.2 Background . . . 21
3.3 Experiments . . . 25
3.4 Results . . . 32
3.5 Conclusions and Future work . . . 42

4 Nocturne: Multi-Agent Driving Benchmark **44**
4.1 Introduction . . . 44
4.2 Related Work . . . 46
4.3 Benchmark Construction . . . 49
4.4 Experiments Setup . . . 53
4.5 Results and Analysis . . . 54
4.6 Conclusion . . . 57

5 Deep RL Traffic Smoothing Controllers **59**
5.1 Introduction . . . 59
5.2 Problem Formulation . . . 61
5.3 Controller Design . . . 63
5.4 Results . . . 67
5.5 Conclusion . . . 70

II Field Validation: 100-Vehicle Highway Deployment **72**

6 Deploying Traffic Smoothing Cruise Controllers **73**
6.1 Introduction . . . 73
6.2 Related Work . . . 75
6.3 Training Set . . . 76
6.4 Method . . . 79
6.5 Results . . . 81
6.6 Conclusions and Future Work . . . 85

7 100 RL-controlled AVs Highway Field Experiment **87**
7.1 Introduction and Related Work . . . 87
7.2 Design of the Controller Architecture . . . 91
7.3 Design of Controller Components . . . 99
7.4 Candidate Controller Selection . . . 108
7.5 Open Road Field Operational Test . . . 110
7.6 Results . . . 115
7.7 Conclusion . . . 116

8 Neural Finite Volume Methods for Hyperbolic PDEs **129**
8.1 Introduction . . . 129
8.2 Related work . . . 132
8.3 Prerequisites and Notations: Finite Volume Methods . . . 133
8.4 Our method: Neural Finite Volume (NFV) . . . 134
8.5 Experiments . . . 135
8.6 Application: Modeling Large-Scale Experimental Field Data using NFV . . . 139
8.7 Conclusion . . . 142

9 Conclusion and Perspectives **144**
9.1 Faster and More Accurate Simulations . . . 145
9.2 Improved Traffic Prediction . . . 146
9.3 Vehicle Communication . . . 147
9.4 Hardware Integration . . . 147
9.5 Scaling Challenges . . . 148

Bibliography **149**

A Bottleneck Control **171**
A.1 Training Parameters 171
A.2 Experiment Details 171
A.3 Reproducibility and Experimental Details 173
A.4 Feedback Controller Sweep Parameters 173

B Nocturne **175**
B.1 Experiment Details 175
B.2 Zero-shot Coordination 177
B.3 Ablation of the Targets 177
B.4 Vehicle Model 178
B.5 Calculating Infeasible Goal Statistics 179
B.6 Computing the Simulator Speed 179

C Neural Finite Volume **181**
C.1 Finite Volume Methods 181
C.2 Variants of LWR 182
C.3 More General PDE Models 184
C.4 I-24 Experimental Dataset: Density Extraction, Training, and Evaluation . . 185
C.5 Experiment Details 189

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisors, Alexandre Bayen and Amaury Hayat, for their invaluable guidance, encouragement, and patience throughout my PhD. They took a chance on me, for which I am extremely grateful, and have since gone above and beyond in supporting my research and professional growth. Their technical depth and research expertise have been a constant source of inspiration, and I am profoundly appreciative of all I have learned from them. I would also like to thank Francis Bach for introducing me to Alexandre during my time at ENS Paris-Saclay, and Maria Laura Delle Monache, Claire Tomlin, and Sergey Levine for serving on my qualifying exam and dissertation committees, and for all their thoughtful advice and feedback.

Massive thanks go to Eugene Vinitsky for being an inspiring mentor and collaborator since my very first days in the lab. We have co-authored numerous papers together in the wonderful field of reinforcement learning, and I am deeply grateful for these opportunities and for the external collaborations he made possible. Eugene not only introduced me to research and to the field itself, but also showed me what it means to be a great and respectful mentor and researcher, sharing values that I will carry forward.

Naturally, my PhD wouldn't have been possible without the many rewarding collaborations with talented students and researchers. I am especially grateful to Eugene Vinitsky, Kathy Jang, Adit Shah, Zhe Fu, Hossein Nick Zinat Matin, Alexi Canesse, Matthew Nice, Kala Agbo Bidi, Xiaomeng Yang, Max Rudolph, Sobhan Mohammadpour, Samuel Sokota, Gabriele Farina, Ashwin Dara, Ashish Pandian, and many others. It has been a true pleasure working with you all. I am also thankful for the chance to collaborate with industry partners at Nissan, Toyota, General Motors, and Meta, with special appreciation to Sharon Hornstein, Brandon Amos, and Jakob Foerster. Finally, my thanks go to Jean-Michel Coron, George Bastin, and Benedetto Piccoli for all the insightful discussions and collaborations.

I had the privilege of being part of the CIRCLES project and the MegaVanderTest, where I deployed my own reinforcement learning controllers on a hundred cars, at a scale usually not achievable during a PhD. It was a truly unique and memorable experience, and I am deeply grateful for the opportunity. I owe special thanks to Jonathan Lee for his outstanding leadership. His expertise and dedication single-handedly unified the efforts of over 50 researchers and steered the project to success. I am deeply thankful to the PIs for their trust and support: Jonathan Sprinkle, Benedetto Piccoli, Dan Work, Benjamin Seibold, Alexandre Bayen, and Maria Laura Delle Monache. Special thanks go to Mostafa Ameli for many engaging discussions, and to the rest of the RL team – Kathy Jang, Adit Shah, and Eugene Vinitsky – for the intense but rewarding MVT week and all the work leading up to it.

My time in the lab has been filled with incredible people who made this journey far more fun, inspiring, and memorable. I am deeply grateful for your friendship and support. Big thanks to Fangyu Wu for the memes, the deep conversations at Aki, and the road trips with Alben Bagabaldo. I'm thankful to Zhe Fu for teaching me tennis and Mandarin, helping me raise my spicy food tolerance, and sharing concerts, and I've enjoyed many long conversations with Arwa AlAnqary and Hossein Nick Zinat Matin. I'm also grateful to Kathy Jang for

the late-night paper crunches and to Han Wang for always blowing my mind. I also want to thank Joy Carpio, Sulaiman Almatrudi, Aboudy Kreidieh, Adit Shah, Zihe Liu, Jordan Kam, Mladen Čičić, Yasmine Marani, Elisa Gaetan, and the French gang – Valentin Denée, Zakaria Baba, Adrien Bindel, Martin Drieux, Alexi Canesse, and Victor Morand – for all the conversations, shared meals, good times, and a memorable trip to Hawaii.

When I first joined the lab for an internship, I was lucky to meet Alexander Keimer and Tanya Veeravalli, who showed me around and made the experience memorable with insightful conversations, laughter, and countless chess games. I'm also grateful to Ashkan Yousefpour, who brightened the office with his energy. Thanks to Yiling You, Jiayi Li, Yashar Farid, Shu-Xia Tang, Jessica Lazarus, Théophile Cabannes, Fang-Chieh Chou, Cathy Wu, and Marsalis Gibson for the great conversations and making the lab an enjoyable place. I'll remember the lab dinners we shared, the great conversations, hikes, and the trip to Yosemite.

Being at Berkeley has been a privilege, from countless campus walks and unexpected nighttime encounters with deer to enjoying the piano rooms, wonderful Russian lessons with Oksana Willis, and excellent courses from many professors. I especially thank Kameshwar Poolla and Murat Arcak for teaching me what I know about classical control, and Gireeja Ranade for inspiring lectures. The squirrelful campus, the melodies of the Campanile, and the hikes up the hills above campus, which ended with a breathtaking view of the bay, especially at sunset, are memories that will stay with me.

I also had the opportunity to teach CS70 for two semesters, my first experience with a class of many hundreds of students. I am grateful to professors Satish Rao, Avishay Tal, Alistair Sinclair, and Sanjit Seshia for their trust and guidance, and to the entire teaching staff, especially Alec Li, along with Jennifer Zhao, Andrew Klearman, Sriram Srivatsam, and Jason Jiang, for making it an enjoyable and rewarding experience.

Many thanks to Shirley Salanio, Helen Bassham, Annie Yeh, and Judy Smithson for their invaluable help in navigating administrative matters and for the friendly conversations.

I am grateful for funding from CNRS and from the UC Berkeley Fellowship. Thanks to AWS, Azure, Jean-Zay, and the Savio cluster at Berkeley for the computational resources that made this work possible. I am also grateful to the broader research community for open-source tools such as SUMO, PyTorch, the Waymo Motion dataset, SB3, Aleksei Petrenko's SampleFactory, Costa Huang's cleanrl, and Joseph Suarez's PufferLib, which I have greatly enjoyed using. Thanks to Andrej Karpathy, Lilian Weng, and many others for their educational blogs and videos, and to David Silver and Sergey Levine for their lectures that introduced me to RL. Last but not least, thanks to my reinforcement learning agents for tirelessly exploring, exploiting, and occasionally engaging in elaborate shenanigans, often doing the exact opposite of what I had in mind, and making the journey all the more unpredictable and fun.

Finally, I am deeply grateful to all of my family, and especially to my parents, Francis and Christine, for their unwavering belief in me and constant support, even without always fully understanding the details of my work, and for trusting me to pursue this PhD far from home, across the Atlantic. Your encouragement made this journey possible. Merci pour tout. To my cat Maya, thank you for always greeting me when I came home. And to Stasya, thank you for your patience, encouragement, and endless support throughout this adventure.

Chapter 1

Introduction

1.1 Motivation

Autonomous vehicles (AVs) have rapidly progressed from research prototypes to real-world deployment, with various levels of automation now in widespread use. Most new cars feature Level 1 or Level 2 advanced driver-assistance systems, which include features like adaptive cruise control, lane keeping assist, and automated emergency braking. These systems provide partial automation, where the vehicle can assist with driving tasks but still requires the human driver to remain attentive and ready to take control. Tesla's Autopilot, introduced in 2015, and GM's Super Cruise, launched in 2017, are examples of Level 2 systems where the vehicle can steer, accelerate, and brake, but the human driver must remain attentive and ready to take control. In 2023, Mercedes-Benz's Drive Pilot became the first certified Level 3 system approved for use in the United States. It allows the car to fully manage driving under limited highway conditions, while the driver can disengage from active control but must be able to resume driving when prompted. At Level 4, Waymo (a subsidiary of Alphabet) operates fully driverless robotaxi services in cities such as San Francisco and Phoenix, with no human driver present during the ride, while Cruise (owned by General Motors) has offered similar services in select cities. Other companies, including Aurora, Zoox (owned by Amazon), and Motional (backed by Hyundai and Aptiv), are actively testing Level 4 autonomous systems in urban and highway settings. Traditional automakers continue to invest heavily in autonomous technology, and regulatory frameworks are gradually evolving to support broader deployment. However, most current efforts focus on mimicking human driving behavior. This approach misses the opportunity to harness the unique capabilities of AVs to improve traffic on a larger scale. This dissertation mainly investigates how AVs can be controlled more intelligently and cooperatively to reduce traffic congestion, improve energy efficiency, and enhance safety in various traffic scenarios.

Highways provide an ideal setting for studying this problem and will be the primary focus of this thesis. They handle massive volumes of traffic, are economically critical, and have relatively straightforward geometry. Yet despite this apparent simplicity, highways frequently

suffer from self-generated instabilities, commonly called *phantom jams* or *stop-and-go waves*, that form without any clear cause like an accident or lane closure. These waves instead emerge from small disruptions in human driving, such as a slightly delayed brake or an overreaction to a lane change, that propagate backward and grow in intensity. Phantom jams are not only frustrating, they come with real costs: wasted fuel, lost time, increased carbon emissions, and a higher risk of collisions. While infrastructure-based solutions like ramp metering (which uses traffic signals to control the rate at which vehicles enter the highway) and variable speed limits can help, they often fall short when traffic instability emerges spontaneously within a highway segment rather than at its entry or exit points.

A promising alternative is to view AVs as distributed actuators: vehicles that are not just autonomous, but actively help regulate traffic flow from within. The idea is that even a small fraction of well-controlled AVs, mixed into regular traffic, could smooth out stop-and-go waves and reduce congestion for everyone. These AVs respond more precisely and consistently than human drivers, helping to dampen the small disturbances that typically grow into larger traffic jams. Unlike traditional solutions that rely on building new infrastructure, this approach could be implemented through software updates to existing adaptive cruise control systems, making it far more scalable and cost-effective. The key challenge is to design control strategies that remain effective under real-world conditions, where sensing is noisy, visibility is limited, and vehicles cannot always communicate directly with each other.

Designing control strategies that are both effective and robust for autonomous vehicles in real-world traffic is a deeply complex problem. Traditional methods from control theory, which rely on carefully tuned rules or mathematical models, can struggle to handle the complexity of highway traffic. Highways are highly dynamic systems involving many interacting vehicles, where conditions are constantly changing and each vehicle has only a limited view of its surroundings. These systems are nonlinear, uncertain, and partially observable, making them challenging to control with fixed, rule-based approaches. Classical control methods can lack the flexibility to adapt to the unpredictable nature of traffic flow, where small disturbances can lead to significant disruptions. This limitation highlights the need for more adaptive and intelligent control strategies that can respond to the dynamic and uncertain environment of highway traffic.

Reinforcement learning (RL) offers a promising alternative. Instead of following fixed rules, RL enables a system to learn how to act by interacting with its environment and improving over time through trial and error. This approach has shown impressive results in a variety of complex settings, including mastering the games of chess and Go with AlphaZero, training robots to manipulate unfamiliar objects, optimizing computer chip designs, and supporting the reasoning abilities of large language models. These successes show that RL can discover effective strategies in environments that are large, unpredictable, and constantly changing, much like highway traffic. This thesis focuses on using deep RL to develop AV control strategies that can adapt to real-world conditions and help improve traffic flow, not just for individual vehicles but for the system as a whole.

Developing these control strategies involves solving fundamental challenges in multi-agent coordination, realistic simulation, and safe deployment. Multiple autonomous vehicles

must learn to coordinate their actions implicitly using only local sensing, without direct vehicle-to-vehicle communication. Training such controllers requires simulation environments that capture the complexity of highway traffic dynamics while remaining computationally efficient enough for reinforcement learning. Most importantly, the learned policies must also generalize beyond simulation to operate safely on production vehicles in unpredictable traffic conditions. To address these challenges, we begin by designing and testing algorithms in controlled simulation environments, establishing multi-agent traffic frameworks, and building scalable training infrastructure. We then move toward deployment, starting by validating the algorithms with single-vehicle tests and culminating in a large-scale field experiment: a 100-vehicle experiment on Interstate 24 in Tennessee. In this live traffic demonstration, deep RL traffic-smoothing controllers run on 100 connected and automated vehicles during rush hour, achieving measurable reductions in stop-and-go waves and thus providing empirical validation of learning-based methods for mixed-autonomy traffic control.

1.2 Thesis Organization

The thesis is organized as follows. First, Chapter 2 reviews the theoretical foundations for traffic flow theory, reinforcement learning, and mixed autonomy control. The work in this dissertation is then separated into two main parts leading up to the 100-vehicle field experiment. Part I focuses on developing the algorithmic foundations through simulation studies, while Part II transitions to field validation, beginning with small-scale tests and culminating in the full 100-vehicle deployment.

In Part I, Chapter 3 develops a multi-agent learning framework showing how multiple decentralized AVs can coordinate using only local sensing without vehicle-to-vehicle communication to optimize traffic flow in a road bottleneck scenario. Then, Chapter 4 introduces Nocturne, a fast and scalable simulation infrastructure for training controllers to cooperate in complex multi-agent driving scnearios based on real data. After that, Chapter 5 presents a comprehensive study where real-world trajectory data from highways is collected and used to train controllers. This approach focuses on capturing realistic highway wave dynamics to smooth traffic waves and reduce fuel consumption.

In Part II, Chapter 6 presents an integral pipeline from simulation to deployment, including controller training, safety protocols and zero-shot policy transfer to production vehicles. This work demonstrates that learned behaviors can maintain their intended performance when transferred to production vehicles under variable traffic conditions. Chapter 7 then presents the culmination of this research: a fleet of 100 connected and automated vehicles operating in live highway traffic in one of the largest flow smoothing traffic experiments. It describes how the single-controller approach is scaled to this 100-vehicle field experiment, integrating live traffic data with RL-based AV control, to achieve wave smoothing within congested rush-hour traffic. Additionally, Chapter 8 explores enhanced traffic prediction methods that combine partial differential equations, modelling traffic conservation laws, with physics-based neural networks, providing methodological foundations that could improve the prediction capabilities

of future traffic control systems. Finally, Chapter 9 concludes the thesis by summarizing the key contributions and highlighting future directions for research in this area.

1.3 Acknowledgement of Contributions

The work presented in Chapters 3 through 8 of this dissertation is based on collaborations with my co-authors, and has been published in peer-reviewed venues for the most part. While I was a primary contributor to each of these works, they were team efforts. The co-authors and corresponding papers are listed at the beginning of each chapter. The ideas, experiments, and results presented in these chapters are original contributions.

Chapter 2

Background

This chapter lays out the key concepts and theoretical background that support the methods developed in this dissertation. We start by reviewing classical traffic flow theory, focusing on how density, flow, and speed interact to shape traffic patterns and congestion. Next, we look at microscopic car-following models and explore how string stability affects the way disturbances move through traffic, giving rise to stop-and-go waves and associated energy losses. Building on this, we introduce the idea of mixed-autonomy traffic control, where autonomous vehicles act as mobile actuators to help shape traffic flow. Finally, we give an overview of deep reinforcement learning methods, including policy optimization and the challenges of coordinating multiple autonomous agents to improve system-wide performance. Note that this chapter is not meant to provide an exhaustive introduction of each topic, but rather to give a high-level overview and point to relevant references; additional background is introduced in later chapters as needed.

2.1 Traffic Flow Theory

Traffic flow theory provides the mathematical foundation for understanding how vehicles move and how traffic congestion emerges [1]. It focuses on three fundamental macroscopic quantities that describe the state of traffic: density ρ (vehicles per unit length), flow q (vehicles per unit time), and speed v (distance per unit time). These quantities are related through the conservation equation $q = \rho v$, which forms the basis for analyzing traffic dynamics.

The relationship between flow and density is captured by the *fundamental diagram* $q = f(\rho)$, where f is typically a concave function of density that is maximized at a critical density ρ_c where the road reaches its throughput capacity $q_{\max}$. This critical point divides traffic into two distinct regimes: free-flow conditions where $\rho < \rho_c$ and vehicles can maintain their desired speeds, and congested conditions where $\rho > \rho_c$ and interactions between vehicles force reduced speeds.

The fundamental diagram also defines the jam density $\rho_{\max}$, the maximum possible vehicle density, at which flow approaches zero. While the fundamental diagram can be approximated

empirically using traffic data from a specific road, common simplified models include the Greenshields model $f(\rho) = v_{\max}\rho\,(1 - \rho/\rho_{\max})$ [2], illustrated in Figure 2.1, where $v_{\max}$ is the road speed limit; and a further simplified triangular model, where flow increases linearly with density until reaching a critical point, then decreases linearly until reaching zero flow.

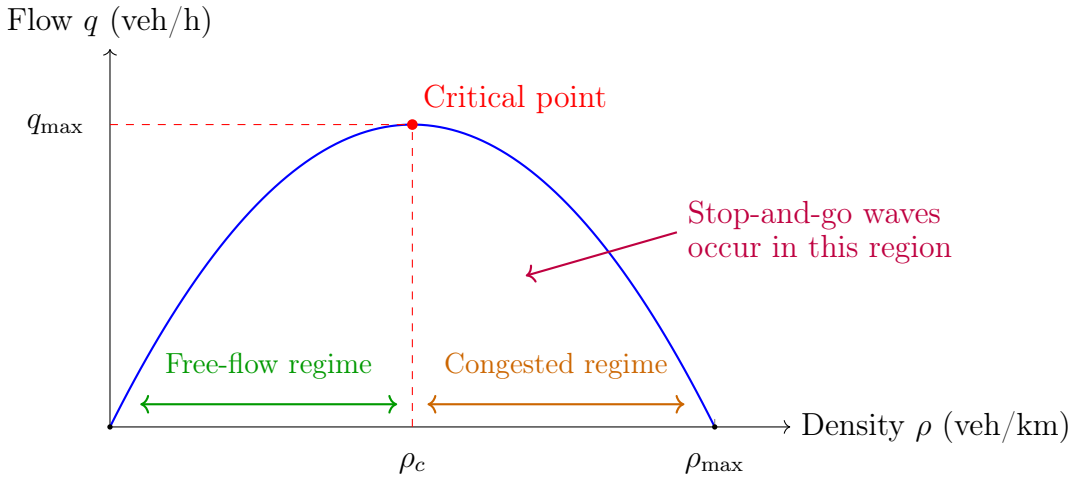


Figure 2.1: The Greenshields fundamental diagram.

Between the critical and jam densities lies the congested regime, where traffic instabilities and wave phenomena naturally emerge. This regime is characterized by the capacity drop phenomenon, where actual throughput falls below the theoretical maximum due to driver behavior and geometric constraints.

Road bottlenecks can represent locations where the fundamental diagram relationship is disrupted, typically occurring at lane reductions, on-ramp merges, or geometric constraints such as curves and grades. At these locations, the upstream capacity exceeds the downstream capacity, creating conditions where congestion can form and propagate backward through the traffic stream. The capacity drop at bottlenecks often reduces throughput to levels substantially below the uncongested capacity, making these locations primary targets for traffic control interventions.

The physical interpretation of these traffic regimes explains why energy efficiency deteriorates in congested conditions. Free-flow traffic allows vehicles to maintain steady speeds with minimal acceleration and deceleration, while congested traffic leads to repeated stop-and-go cycles that waste fuel through inefficient engine operation and increased drag losses. Understanding these energy implications motivates the development of traffic smoothing controllers that can mitigate oscillatory behavior and restore more efficient flow patterns.

2.2 Car-Following Dynamics

Traffic flow can be analyzed at multiple scales, from macroscopic models that treat traffic flow as a fluid to microscopic models that simulate each individual vehicle and interactions between them.

The macroscopic view of traffic flow can be formalized using partial differential equations that treat traffic as a continuum [3]. The Lighthill-Whitham-Richards (LWR) model [4, 5] is a foundational example, combining the conservation equation $\partial_t \rho + \partial_x q = 0$ with the fundamental diagram relationship $q = f(\rho)$ to yield the scalar conservation law $\partial_t \rho + \partial_x f(\rho) = 0$, where the road density ρ is a function of both time and space. This hyperbolic partial differential equation (PDE) describes how traffic density evolves in the spatiotemporal space, naturally capturing the formation and propagation of congestion waves without requiring detailed modeling of individual vehicle interactions. This approach is further developped in Chapter 8 where neural network-based numerical schemes are developped to predict solutions to conservation laws. However, the work and control algorithms developped throughout the remainder of this dissertation operate at the microscopic level, where autonomous vehicles interact with vehicles controlled using car-following dynamics.

Car-following models [6] describe how each vehicle adjusts its acceleration based on the movement of the vehicle directly ahead. Although these models focus on individual vehicles, their interactions can give rise to complex system-level phenomena such as stop-and-go waves, shock propagation, and capacity drops. As such, they provide a foundational framework for understanding how simple local driving behaviors can lead to emergent traffic patterns and instabilities.

The Intelligent Driver Model (IDM) [7] is one of the most widely adopted car-following models, capturing driver behavior through a continuous acceleration function. The IDM determines a vehicle's acceleration based on three primary factors: the desire to reach a preferred speed, the need to maintain safe following distance, and the requirement to avoid collisions with slower-moving leading vehicles. The model's acceleration equation takes the form:

$$a_{\text{IDM}}(v, \Delta v, s) = a_{\max} \left[1 - \left(\frac{v}{v_0} \right)^{\delta} - \left(\frac{s^*(v, \Delta v)}{s} \right)^2 \right]$$

where v is the vehicle's velocity, Δv is the velocity difference (approach rate) to the leading vehicle, s is the bumper-to-bumper distance to the next vehicle, and

$$s^*(v, \Delta v) = s_0 + vT + \frac{v \Delta v}{2\sqrt{a_{\max} b}}$$

represents the desired spacing. The model parameters include the desired speed v_0, minimum time headway T, maximum acceleration $a_{\max}$, comfortable deceleration b, exponent δ (typically $\delta = 4$) and minimum spacing s_0. It aims to reach the desired free-flow speed v_0 when the gap s to the leading vehicle is large, while smoothly decelerating when approaching a slower car (i.e., large Δv) or when the gap becomes small. This results in a smooth and realistic

representation of driver behavior that captures key traffic dynamics using relatively few tunable parameters. Other widely studied car-following models include the Gipps model [8], the Optimal Velocity Model (OVM) [9], and the Full Velocity Difference Model (FVDM) [10], each characterized by different assumptions and formulations for how drivers respond to traffic conditions.

2.3 String Stability and Wave Formation

A fundamental concept in analyzing how vehicles follow each other is *string stability* [11]. String stability refers to the ability of a platoon of vehicles to absorb small disturbances, such as one driver tapping the brakes, rather than amplifying them. If each driver overreacts slightly more than the one before, these small changes can grow into large speed fluctuations or stop-and-go traffic waves. A car-following model is said to be string stable if such disturbances get smaller as they move from one vehicle to the next in the upstream direction. Intuitively, string stability can be analyzed using a transfer function, which describes how speed or spacing disturbances from a leading vehicle affect a following vehicle. If $H(j\omega)$ denotes the frequency response from a leader's perturbation to the follower's response, the system is string stable if the gain is less than one for all frequencies, i.e., $\forall\omega,\ |H(j\omega)| < 1$ (sometimes relaxed). A more formal definition can be found in [12]. This condition ensures that the response of each follower remains smaller than the disturbance introduced by the leader, preventing the formation of traffic waves and maintaining smooth flow.

Human drivers typically operate under string-unstable conditions due to reaction delays and conservative gap-keeping behaviors. When drivers maintain large time headways and respond slowly to leader speed changes, small perturbations can grow as they propagate backward through the traffic stream. A single short braking event by one driver can cascade upstream, creating increasingly severe deceleration for following vehicles. This amplification mechanism transforms minor disturbances into the stop-and-go waves commonly observed in highway traffic, periodic oscillations in vehicle speed that propagate upstream. These phantom jams represent one of the most visible and energy-wasteful phenomena in highway traffic, arising spontaneously from the inherent instability of the collective dynamics of human car-following behavior. In contract, externally caused congestion originates from genuine capacity restrictions such as lane closures, on-ramp merges, or bottlenecks, though the resulting wave patterns often exhibit similar propagation characteristics.

Time-space diagrams can provide a useful visualization for understanding wave dynamics, as can be seen in Figure 2.2. The horizontal axis represents time while the vertical axis represents spatial position along the roadway. Each individual vehicle trajectory appears as a curve in this diagram, where the slope of each trajectory line corresponds to the vehicle's speed, which is also color-coded. Steep slopes indicate high speeds (here shown with bright yellow-green colors), while flat or horizontal segments represent slow or stopped conditions (here shown with dark black-red colors). This representation illustrates stop-and-go traffic, where free flow progressively gives way to oscillatory wave patterns that propagate upstream

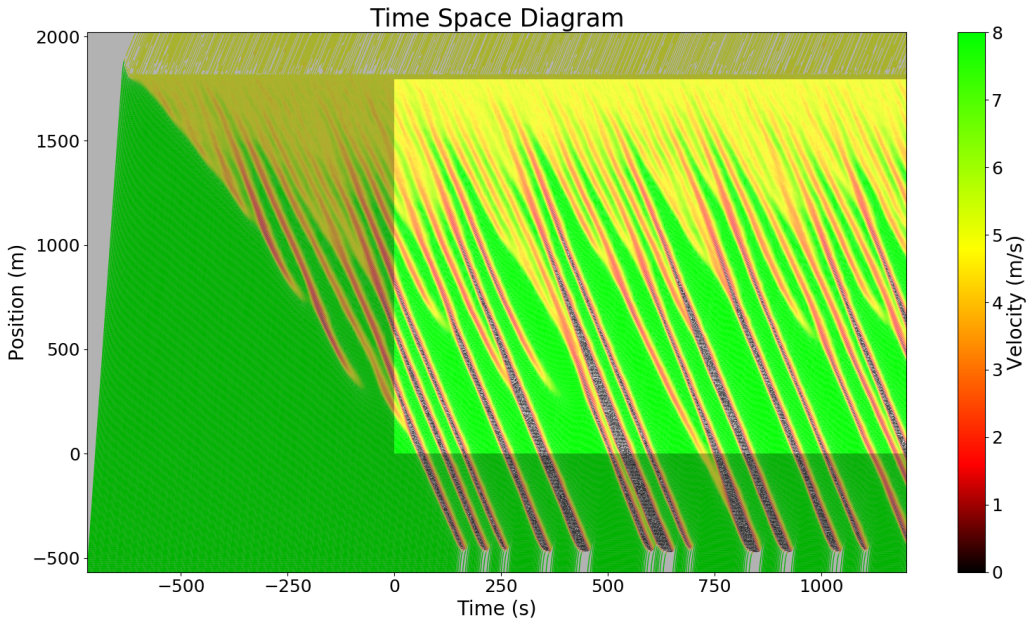


Figure 2.2: Example time-space diagram for illustrative purposes. A simulation of the I-210 highway was conducted in the microscopic simulator SUMO [13] with a continuous influx of vehicles. The grayed-out area corresponds to a padding time while the system warms-up, as well as the edges of the highway network. The downward-pointing dark lines correspond to stop-and-go waves.

in the traffic flow, at speeds given by the fundamental diagram of the road.

The energy implications of traffic waves are substantial. Each oscillation cycle forces vehicles through repeated acceleration and deceleration phases, dissipating kinetic energy during braking that must be restored through fuel consumption during subsequent acceleration. Modern internal combustion engines are most efficient when running at steady speeds, but they use much more fuel during acceleration phases. The stop-and-go pattern characteristic of traffic waves places engines in precisely these inefficient operating regimes. These energy penalties compound across the traffic stream, as multiple vehicles experience simultaneous oscillations that amplify the total fuel consumption beyond what would be required for equivalent distance traveled at steady speeds.

Consequently, understanding the microscopic instability mechanisms leading to the formation of phantom waves is key to design autonomous vehicle controllers that can disrupt this amplification process by absorbing perturbations instead of growing them. As such, most of this works considers simulations where human-driven vehicles are modeled using the IDM, with a set of parameters that make the model string-unstable to replicate the imperfections of human driving. As such, the model is able to replicate stop-and-go wave patterns observed in real-world traffic. This setup will allow us to introduce controlled perturbations and evaluate

the effectiveness of our proposed autonomous vehicle control algorithms at dampening the perturbations and restoring smooth flow, despite an otherwise unstable traffic system.

2.4 Mixed Autonomy Traffic Control

As more and more automated and autonomous vehicles are gradually introduced onto roads, they begin to share space with human-driven cars, creating a mixed autonomy traffic regime. Unlike fully autonomous systems where all vehicles can coordinate through communication protocols, mixed autonomy requires autonomous vehicles to interact with conventional drivers. This provides an opportunity to improve traffic by using a small number of controllable vehicles as mobile actuators that help shape traffic flow from within. Depending on the situation and the goal, these vehicles could smooth stop-and-go waves to reduce congestion and energy use, manage merging at intersections, bottlenecks, or ramps, and adjust speeds dynamically to prevent capacity drops. These benefits can be realized even during the early stages of autonomous vehicle adoption, long before widespread deployment.

The FLOW simulation platform [14] provides a foundational framework for studying and optimizing mixed autonomy traffic scenarios. FLOW integrates traffic microsimulators such as SUMO [13] with reinfrocement learning libraries, enabling researchers to train autonomous vehicle controllers in traffic networks ranging from simple rind roads and intersections to complex highway systems. In a ring road scenario, a single autonomous vehicle trained in FLOW learned to dampen stop-and-go waves by driving slightly differently and maintaining enough headway to absorb perturbations rather than propagate them. This led to fuel savings for all vehicles on the ring. The key insight is that even a small number of autonomous vehicles can generate system-wide improvements when their control strategies are designed to optimize collective behavior rather than individual performance.

Building on these simulation results, the Arizona State University ring road experiment [15] provided empirical validation. The study placed a single autonomous vehicle among 21 human drivers on a circular track, replicating the stop-and-go waves seen in highway traffic. The autonomous vehicle used a simple control strategy that maintained consistent spacing and avoided unnecessary acceleration and braking. The results showed that this single vehicle could eliminate traffic oscillations and reduce total fuel consumption by up to 40% compared to the all-human baseline. The experiment demonstrated that traffic instabilities are fragile and can be disrupted by a well-controlled vehicle.

These findings raise important questions about scalability to highway networks, where traffic patterns are more complex and vehicle interactions occur across multiple lanes. Highways introduce additional challenges, including lane changes, on-ramp merging, and varying penetration rates of autonomous vehicles. Despite these complexities, the core mechanism of wave absorption still applies: autonomous vehicles can mitigate phantom traffic jams by breaking string-unstable feedback loops and maintaining smoother car-following behavior than human drivers.

This leads to the central question of this thesis: can a small penetration rate of just 1 to 5 percent autonomous vehicles on highways reduce traffic oscillations and fuel consumption, not only for themselves but for all vehicles on the road? Such low adoption rates are plausible in the near future. While higher penetration is expected to yield greater benefits, the relationship is not linear. As this dissertation demonstrates, significant improvements are already achievable at low levels of adoption.

Deploying mixed autonomy control algorithms in practice also presents many challenges, particularly in sensing and communication. Unlike ring road experiments that could leverage global state information, highway scenarios must operate using local sensor data such as radar-based distance and relative speed. Fortunately, many vehicles equipped with adaptive cruise control already possess the necessary sensing capabilities, making the transition toward traffic-smoothing controllers more attainable.

Although vehicle-to-vehicle (V2V) communication could further enhance coordination, the control strategies developed in this dissertation do not depend on it. Instead, they emphasize decentralized approaches that rely only on local observations. This improves deployability on existing vehicles and enables broader use across different road settings. When available, macroscopic traffic data obtained from infrastructure or other vehicles can still supplement local sensing. A further advantage of decentralized control is its reduced need for road-specific tuning compared to centralized methods.

Ultimately, designing control algorithms for mixed autonomy involves balancing individual vehicle objectives with broader system-level goals. Autonomous vehicles must maintain safety, respect passenger comfort, and contribute to improved traffic flow, all while adhering to social driving norms to avoid unsettling human drivers. This multi-objective problem requires careful formulation of optimization goals, as well as adherence to constraints on observability and actuation.

2.5 Reinforcement Learning

Reinforcement learning addresses the fundamental problem of learning to make good decisions through experience. Consider an autonomous vehicle navigating traffic: at each moment, it must decide whether to accelerate, brake, or maintain speed based on current conditions. The vehicle receives feedback through various signals (smoother traffic flow, reduced fuel consumption, or passenger comfort) but this feedback often comes with delay and reflects the cumulative effect of many previous decisions. The challenge lies in learning which actions lead to desirable long-term outcomes when the consequences of individual actions are not immediately apparent.

This learning paradigm contrasts with supervised learning, where correct answers are provided for each input, by requiring an agent to discover strategies through trial and error while balancing exploration of new possibilities with exploitation of currently known good actions. The autonomous vehicle must learn that maintaining consistent spacing can prevent

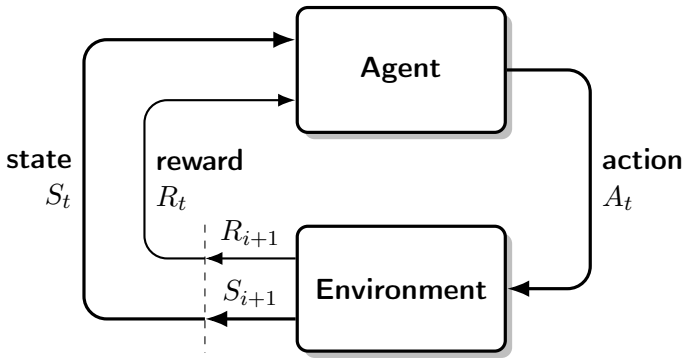


Figure 2.3: The reinforcement learning agent-environment interaction cycle. At each time step t, the agent observes the current state S_t, selects an action A_t, and receives a reward R_{t+1} along with the next state S_{t+1} from the environment.

traffic waves from forming, even though this strategy may initially seem counterintuitive or require temporary speed reductions.

The mathematical foundations of reinforcement learning trace back to dynamic programming work by Bellman in the 1950s [16] and early psychological studies of animal learning [17], with the modern computational framework emerging through convergence of optimal control theory, psychology, and artificial intelligence research during the 1980s and 1990s [18]. Landmark achievements include temporal difference learning methods such as Q-learning [19] and TD(λ) [20], Deep Q-Networks (DQN) [21] achieving human-level performance on Atari games, and policy gradient methods [22, 23, 24, 25] enabling breakthroughs in continuous control. More recent successes span AlphaGo's defeat of a Go world champion [26], complex locomotion and manipulation skills [27, 28], helicopter flight [29], robotic manipulation [30], and autonomous driving [31].

Reinforcement learning problems are formalized as Markov Decision Processes (MDPs), defined by the tuple

$$(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma),$$

where $\mathcal{S}$ represents the state space, $\mathcal{A}$ the action space, $\mathcal{P}(s'|s,a)$ the transition dynamics, $\mathcal{R}(s,a)$ the reward function, and $\gamma \in [0,1)$ the discount factor. At each time step t, the agent observes state $s_t \in \mathcal{S}$, selects action $a_t \in \mathcal{A}$ according to policy $\pi(a|s)$, receives reward $r_t = \mathcal{R}(s_t, a_t)$, and transitions to state $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$, as illustrated in Figure 2.3. The Markov property ensures that transition probabilities depend only on the current state and action, not on the complete history. Formally, this means $\mathcal{P}(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \ldots) = \mathcal{P}(s_{t+1}|s_t, a_t)$. This memoryless property simplifies analysis while capturing many practical scenarios, though it requires careful state design to ensure all relevant information is included in the current state representation.

In many practical applications however, agents cannot observe the complete state of the environment, leading to Partially Observable Markov Decision Processes (POMDPs)

being used. A POMDP extends the MDP framework with an observation space Ω and an observation function $\mathcal{O}(o|s,a)$ that defines the probability of receiving observation o after taking action a and transitioning to state s. Unlike MDPs where the agent directly observes the current state s_t, in POMDPs the agent receives observations o_t that provide only partial information about the underlying state. This formulation captures realistic scenarios where sensors have limited range or accuracy, information is noisy or incomplete, or the full system state is simply too large to observe entirely.

A policy can be either deterministic, where $a = \pi(s)$ specifies a single action for each state, or stochastic, where $a \sim \pi(a|s)$ defines a probability distribution over actions given the current state. Stochastic policies are particularly useful in partially observable environments or when exploration is needed, while deterministic policies are often sufficient for well-defined control problems once learning is complete.

The agent's objective is to learn a policy π that maximizes the expected cumulative discounted reward:

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right],$$

where the expectation is taken over trajectories $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$ generated by following policy π. The discount factor γ determines how much future rewards are valued relative to immediate ones, with γ close to 0 prioritizing immediate rewards and γ close to 1 giving nearly equal weight to all future rewards.

This optimization problem can be approached through value functions that estimate expected returns from particular states or state-action pairs. The state value function

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_t \mid s_0 = s\right]$$

represents the expected return starting from state s and following policy π, while the action value function

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_t \mid s_0 = s, a_0 = a\right]$$

represents the expected return from taking action a in state s and subsequently following π.

These value functions satisfy recursive Bellman equations that express the relationship between the value of a state and the values of its successor states, as per the nature of dynamic programming. The optimal policy $\pi^* = \arg\max_\pi J(\pi)$ maximizes expected return for all states, leading to optimal value functions $V^* = V^{pi^*}$ and $Q^* = Q^{\pi^*}$ that satisfy the Bellman optimality equations:

$$V^*(s) = \max_a\left[\mathcal{R}(s,a) + \gamma\sum_{s'}\mathcal{P}(s'|s,a)V^*(s')\right],$$

$$Q^*(s,a) = \mathcal{R}(s,a) + \gamma\sum_{s'}\mathcal{P}(s'|s,a)\max_{a'}Q^*(s',a').$$

Reinforcement learning algorithms divide into several primary approaches based on their strategy for finding optimal policies. Value-based methods learn value functions and derive policies from them, with algorithms such as Q-learning [19] using temporal difference updates to estimate $Q^*(s, a)$ and then extract the greedy policy $\pi^*(s) = \arg\max_a Q^*(s, a)$. These methods work well for discrete action spaces but face challenges with continuous actions. Policy-based methods parameterize the policy directly as $\pi_\theta(a|s)$ and optimize the parameters θ to maximize expected return, naturally handling continuous action spaces but often suffering from high gradient variance. Actor-critic methods combine both approaches by maintaining separate function approximators for the policy (actor) and value function (critic), with the critic providing variance reduction for policy updates. Modern deep reinforcement learning implements these function approximators using neural networks, enabling the handling of high-dimensional state and action spaces that would be intractable with traditional tabular methods.

Proximal Policy Optimization (PPO) [27] and Soft Actor-Critic (SAC) [28] are among the most widely used algorithms for continuous control problems, valued for their relative stability and straightforward implementation. Other popular used methods include Twin Delayed DDPG (TD3) [32], Deep Deterministic Policy Gradient (DDPG) [24], Advantage Actor-Critic (A2C/A3C) [33], and Deep Q-Networks (DQN) for discrete action spaces [21]. However, state-of-the-art algorithms are often domain-specific: for instance, model-based online control often employs methods like DreamerV3 citehafner2024masteringdiversedomainsworld, offline reinforcement learning relies on approaches such as Implicit Q-Learning (IQL) [34] or Conservative Q-Learning (CQL) [35], and large language model fine-tuning frequently uses Gradient-Regularized PPO (GRPO) [36] or other variants of PPO. These algorithms should be viewed as widely used baselines rather than universal solutions, with optimal choices depending on specific problem characteristics and constraints.

2.6 Policy Gradient Methods

Policy gradient methods provide a direct approach to optimizing parameterized policies by using gradient ascent on the expected return. Unlike value-based methods that learn value functions and derive policies indirectly, policy gradient algorithms parameterize the policy as $\pi_\theta(a|s)$ and directly optimize the policy parameters θ to maximize the objective function $J(\pi_\theta)$. This direct optimization approach is particularly valuable for continuous control problems where the action space is high-dimensional or where stochastic policies are beneficial, making policy gradient methods well-suited for autonomous vehicle control applications.

The mathematical foundation for policy gradient methods rests on the policy gradient theorem [37], which establishes how to compute gradients of the expected return with respect to policy parameters. For a parameterized policy π_θ, the gradient of the expected return can be expressed as:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim d^{\pi_\theta}, a\sim\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a)\right],$$

where d^{π_θ} represents the stationary state distribution under policy π_θ, and $Q^{\pi_\theta}(s, a)$ is the action value function.

This expression enables Monte Carlo estimation of the gradient by sampling full episodes using the policy, which leads to the REINFORCE algorithm [22], the most basic implementation of the policy gradient theorem. In practice, the Q-value is typically approximated by the observed return $G_t = \sum_{t'=t}^{T} r_{t'}$, yielding an unbiased but high-variance gradient estimate. The variance arises because G_t depends on the entire future trajectory, making the resulting gradients noisy and potentially slowing convergence.

Actor-critic methods address the variance problem by combining policy gradients with learned value function baselines. The architecture consists of two components: an actor network that learns the policy $\pi_\theta(a|s)$, and a critic network that learns to estimate value functions, typically either the state value function $V_\phi(s)$ or the action value function $Q_\phi(s, a)$. The advantage function $A^{\pi_\theta}(s, a) = Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s)$ plays a central role by measuring how much better action a is compared to the average action in state s. Using the advantage function instead of $Q^{\pi_\theta}(s, a)$ in the policy gradient typically reduces variance while maintaining the unbiased gradient estimate:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim d^{\pi_\theta}, a\sim\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) A^{\pi_\theta}(s, a)\right].$$

The actor updates follow the policy gradient direction weighted by advantage estimates, while the critic updates minimize the temporal difference error between predicted and observed returns. In practice, the advantage function is often approximated using temporal difference methods, such as $A(s_t, a_t) \approx r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$, where the critic provides the baseline that reduces gradient variance.

Proximal Policy Optimization (PPO) [27] represents a modern refinement of actor-critic methods that addresses stability challenges inherent in policy gradient optimization. Traditional policy gradient methods can suffer from destructively large policy updates that degrade performance, particularly when the policy changes substantially between data collection and parameter updates, i.e. when the policy is updated using stale data. The problem arises because policy gradient methods typically collect a batch of experience using the current policy, then perform multiple gradient steps on this fixed dataset, potentially leading to policy updates that are too aggressive.

PPO introduces a clipped objective function that constrains the magnitude of policy updates by limiting how much the policy can change from the behavior policy used to collect experience. The key insight is to define a probability ratio between the new and old policies:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)},$$

and then clip this ratio to prevent large policy changes. The PPO clipped objective function becomes:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min\left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right],$$

where $\hat{A}_t$ is the advantage estimate at time t, and the clipping parameter ϵ (typically $\epsilon = 0.2$) controls the permitted deviation range. Intuitively, this clipping mechanism ensures that the objective function is pessimistic in the sense that it only allows policy improvements when they are conservative, preventing overly aggressive updates that could make learning unstable.

PPO also incorporates additional elements that improve practical performance, including value function clipping to maintain consistency between policy and value updates, and entropy bonuses to encourage exploration. The complete PPO objective combines the clipped policy loss with a value function loss and an entropy term:

$$L(\theta) = L^{\text{CLIP}}(\theta) - c_1 L^{\text{VF}}(\theta) + c_2 S[\pi_\theta](s_t),$$

where L^{VF} is the value function loss, $S[\pi_\theta]$ is an entropy bonus, and c_1, c_2 are hyperparameters balancing these terms.

For the traffic control applications developed in this dissertation, we primarily use Proximal Policy Optimization (PPO) due to its stability and suitability for continuous control problems. PPO's ability to handle continuous action spaces and constrain policy updates makes it well-suited for training autonomous vehicle controllers that must balance multiple objectives.

2.7 Multi-Agent Learning Challenges

Extending reinforcement learning from a single-agent to a multi-agent setting introduces significant complexity. In contrast to the relatively stable environment of single-agent learning, each agent in a multi-agent system operates in a non-stationary environment shaped by the concurrent learning and adaptation of other agents. This leads to a moving target problem, where the optimal policy for one agent depends on the policies of others, which are themselves evolving during training.

This non-stationarity violates the Markov property assumed in standard theoretical guarantees. From the perspective of a single agent, the hidden state implicitly includes the policies of other agents, making the learning problem partially observable and more difficult to optimize. As a result, experience replay buffers may contain transitions drawn from many different joint policy configurations, destabilizing learning. While the entire multi-agent system can be framed as a centralized single-agent problem by treating joint actions as tuples, this quickly becomes intractable in practice. Instead, this dissertation adopts a decentralized framework, where a shared policy is learned and deployed across all agents.

A second major challenge is credit assignment. In cooperative tasks where agents receive a shared global reward, it becomes difficult to attribute individual contributions. This can lead to uncoordinated or free-riding behavior. On the other hand, assigning local rewards risks encouraging selfish strategies that harm overall system performance. Balancing individual incentives with global objectives is particularly important in systems requiring coordination.

These challenges are especially pronounced in mixed autonomy traffic control. Autonomous vehicles must collaborate to smooth traffic flow or reduce energy consumption, while also

interacting safely with human drivers and acting based on limited observations. Each vehicle's decisions impact others, yet full coordination is difficult due to observability and communication constraints. As the number of agents grows, training becomes more computationally expensive and harder to scale, necessitating careful algorithmic and architectural choices.

Two primary paradigms have emerged to address these issues. Independent learning treats other agents as part of the environment and applies standard single-agent algorithms per agent. This approach is simple and scalable but may suffer from instability when multiple agents adapt simultaneously. It performs better in environments with weak agent interactions or when autonomous agents are sparse relative to human drivers, conditions which are common in our traffic control setting, where this decentralized approach is adopted.

In contrast, centralized training with decentralized execution uses global information during training, such as the full state and other agents' actions, to stabilize learning and improve coordination. This is often done by passing additional information into the critic network. At execution time, however, each agent acts based only on local observations, respecting real-world communication and sensing limitations.

Finally, exploration becomes more difficult in multi-agent systems. Effective strategies often require coordinated exploration, but naive independent exploration can lead to poor coverage of the joint action space. Agents must now reason not only about their own uncertainty but also about the evolving behaviors of others.

In summary, the multi-agent setting introduces core challenges such as non-stationarity, credit assignment, scalability, and exploration, each exacerbated in real-world domains like mixed autonomy traffic, where local information, coordination, and safety must all be balanced.

Part I

Deep Reinforcement Learning for Traffic Control

Chapter 3

Decentralized AVs for Bottleneck Control

This chapter explores the first application of deep reinforcement learning to mixed-autonomy traffic control presented in this thesis. Building on the background presented in Chapter 2, we address a fundamental challenge in traffic engineering: how to prevent capacity drop at highway bottlenecks using decentralized autonomous vehicles. Rather than relying on infrastructure-based solutions like traffic lights or variable speed limits, we investigate whether a small percentage of well-controlled autonomous vehicles can act as mobile actuators to optimize traffic throughput. This work establishes the multi-agent learning framework that underpins the methodological development throughout Part I of this thesis, demonstrating that decentralized control without vehicle-to-vehicle communication can achieve substantial improvements in bottleneck performance across varying penetration rates.

This chapter is based on [38]: Eugene Vinitsky*, Nathan Lichtlé*, Kanaad Parvate, and Alexandre Bayen. "Optimizing Mixed Autonomy Traffic Flow With Decentralized Autonomous Vehicles and Multi-Agent Reinforcement Learning." *ACM Transactions on Cyber-Physical Systems*, 2023.

3.1 Introduction

Highway bottlenecks represent one of the most challenging problems in traffic engineering, where lane reductions and merging zones create conditions for spontaneous congestion formation and capacity drop. As discussed in Chapter 2, the fundamental diagram relationship becomes disrupted at these locations, often reducing throughput to levels substantially below the theoretical capacity. This chapter investigates how autonomous vehicles can be deployed as decentralized mobile actuators to prevent capacity drop and improve bottleneck throughput, establishing the foundation for the multi-agent learning approaches developed throughout this thesis.

Building on the mixed autonomy traffic control concepts introduced in Chapter 2, we demonstrate how level-2 AVs, equipped with standard sensors like cameras and radars, can

be used to improve the throughput of a simplified model of the San Francisco-Oakland Bay Bridge. Rather than requiring infrastructure modifications or centralized coordination, our approach relies on locally acting autonomous vehicles that implicitly coordinate through their shared objective of preventing congestion formation.

We will focus on using AVs to improve the throughput of a lane reduction, a road architecture where the number of lanes suddenly decreases. We will refer to successive lane reductions as a *traffic bottleneck*. Bottlenecks are believed to cause a phenomenon known as *capacity drop* [39, 40] where the inflow-outflow relationship at the bottleneck is initially linear but above some critical inflow value experiences a hysteric transition where the outflow suddenly and sharply drops (see Fig. 3.4 for an example). The imbalance between inflow and outflow leads to congestion and a reduction in the throughput of the bottleneck.

To avoid this reduction, it is necessary to restrict the inflow so that it never exceeds the critical value above which capacity drop occurs. One approach to tackling this is to introduce traffic lights into the network that meter/restrict the inflow [41] but this would require the installation of additional infrastructure. Instead, autonomous vehicles can be used as mobile metering infrastructure, essentially distributed traffic lights, that intelligently select when to meter and when to let the flow continue without restriction.

While there is work characterizing bottleneck control using vehicle-based control, it usually operates in the centralized regime where a single controller outputs commands to all the AVs in the system either via variable speed limits [42, 43, 44, 45] or centrally coordinated platoons [46]. Here we consider the challenging multi-agent problem where each AV operates in a fully decentralized fashion and control is applied at the level of individual vehicle accelerations. The AVs can still coordinate but only implicitly: they can only use common knowledge to decide which AV should go next. While decentralization adds additional difficulty in controller design, the resultant controllers should be realizable using existing cruise control technology and can consequently be implemented without relying on any improvements in vehicle-to-vehicle communication technology.

We investigate the potential impact of decentralized AV control on bottleneck throughput by studying a scaled-down version of the post-tollbooth section (see Fig. 3.1) of the San Francisco-Oakland Bay Bridge. In our scaled version, four lanes reduce to two which then reduce down to one lane (as opposed to 15 to 8 to 5 in the bridge). While the lane numbers differ, the overall road architecture is quite similar as each vehicle goes through two merges. To design the controllers, we will use multi-agent reinforcement learning (MARL). Even at a reduced scale, this problem is a difficult MARL challenge as it incorporates:

- A large number of agents, varying between 20-200 depending on the penetration rate.
- Delayed reward structure. A given vehicle's impact on outflow isn't experienced until many seconds later.
- Challenging credit assignment. The outflow is a global signal and it is difficult to disambiguate whose action led to the improved outcome.

This work tackles this challenging MARL problem and provides some initial characterization of the performance of decentralized control in these settings. The main contributions of this work are:

1. We introduce a challenging new benchmark in multi-agent reinforcement learning.
2. We demonstrate that appropriately chosen multi-agent RL algorithms can be used to design decentralized control policies for maximizing bottleneck throughput.
3. We show that effective control can be performed in the fully local sensing setting where vehicles do not have access to any macroscopic observations.
4. We demonstrate and formalize a challenging problem in open transportation networks where the Nash equilibrium can deviate from the social equilibrium. We introduce a simple trick to make the two equilibria align.
5. We design decentralized feedback control policies and show that, despite extensive tuning, the RL policies sharply outperform our feedback baseline. Additionally, the RL approach is able to equal the performance of a traffic-light baseline.
6. We demonstrate that the resultant control policies can be made robust to variations in the penetration rate.

The rest of the chapter is organized as follows. Section 3.2 provides an introduction to deep RL, off-policy RL methods, car-following models, and *Flow*, the traffic control library used in this work. It also introduces our feedback control baseline and traffic light baseline. Section 3.3 formulates the capacity drop diagrams of our bottleneck and explains the state and action spaces for all the controllers studied herein. Section 3.4 provides a discussion of the results. Finally, Section 3.5 summarizes our work and provides a discussion of possible future research directions.

3.2 Background

3.2.1 Reinforcement Learning

In this section, we discuss the notation and describe in brief the key ideas used in reinforcement learning. Reinforcement learning focuses on maximization of the discounted reward of a *Markov decision process* (MDP) [47] or partially observed Markov decision process (POMDP) in which the agent has restricted access to the true world state. The system described in this chapter solves tasks which conform to the standard structure of a finite-horizon discounted multi-agent POMDP, defined by the tuple $\mathcal{S} \times (\mathcal{A}_0, \mathcal{O}_0, r_0, \gamma_0, T_0) \times \cdots \times (\mathcal{A}_n, \mathcal{O}_n, r_n, \gamma_n, T_n) \times \rho \times P \times \mathcal{Z}$, where $\mathcal{S}$ is a set of world-states, $\mathcal{A}_i$ is a set of actions for agent i, $\mathcal{Z} : \mathcal{S} \times (\mathcal{A}_0 \times \cdots \times \mathcal{A}_n) \to (\mathcal{O}_0, \ldots, \mathcal{O}_n)$ describes how the world state is mapped into the observations

of the agents, $P : \mathcal{S} \times (\mathcal{A}_0 \times \cdots \times \mathcal{A}_n) \to \mathbb{R}_{\geq 0}$ is the transition probability distribution for moving from one set of agent states s to the next set of states s' given the set of actions $(a_0, \ldots, a_n)$, $r_i : \mathcal{S} \times (\mathcal{A}_0 \times \cdots \times \mathcal{A}_n) \to \mathbb{R}$ is the reward function for agent i, $\rho : \mathcal{S} \to \mathbb{R}_{\geq 0}$ is the initial state distribution, $\gamma_i \in (0, 1]$ is the discount factor for agent i, and T_i is the horizon for agent i.

RL studies the problem of how an agent can learn to take actions in its environment to maximize its cumulative discounted reward. Specifically it tries to optimize the expected return $J^\pi = \mathbb{E}_{\rho_0,\ p(s_{t+1}|s_t,a_t)} \left[\sum_{t=0}^{T} \gamma^t r_t \mid \pi(a_t|s_t) \right]$ where r_t is the reward at time t and the expectation is over the start state distribution, the probabilistic dynamics, and the probabilistic controller π. Note that we have temporarily dropped the dependence on agent index for clarity. The goal in RL is to use the observed data from the MDP to compute the controller $\pi : \mathcal{S} \to \mathcal{A}$, mapping states to actions, that maximizes J^π. It is increasingly ubiquitous to parametrize the controller as a neural network. We will denote the parameters of this controller, in this case the neural network weights, by θ and the controller by π_θ. A feed-forward neural network consists of a stacked set of affine linear transforms and non-linearities that the input is alternately passed through. The presence of multiple stacked layers is the origin of the term "deep" reinforcement learning. In this work we will use a shared feed-forward neural network (which we will abbreviate as an MLP - a Multi-Layer Perceptron); each agent will use the exact same controller weights. Details of the architecture are provided in Sec. 3.3.5.

3.2.2 Off-Policy Reinforcement Learning

Here we briefly introduce off-policy reinforcement learning methods in an attempt to clarify some of the difficulties of using single-agent algorithms in multi-agent settings. For a more thorough discussion of the underlying algorithms see [32, 24] and for the particular challenges of multi-agent off-policy algorithms see [48].

Off-policy methods focus on using a buffer of data sampled from the environment to construct the policy. While they can suffer from instability relative to policy gradient methods [49], they tend to be more sample efficient and can often be effectively run on a single CPU. The basic idea is to periodically sample data from the buffer and compute an estimate of the Bellman error

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \left(Q(s_t^i, a_t^i) - r(s_t^i, a_t^i) - \gamma \operatorname*{argmax}_a Q(s_{t+1}^i, a) \right)^2$$

where i indexes a sample from the batch, γ is the discount factor, and Q is the Q-function

$$Q^\pi(s_t, a_t) = \mathbb{E}_\pi \left[\sum_{i=t}^{T} \gamma^{i-t} r_t(s_t, a_t) | s_t, a_t \right]$$

i.e. the expected cumulative discounted reward of taking action a_t and thereafter following policy π (we will use the terms policy and controller interchangeably in this section). The

sum $r(s_t^i, a_t^i) + \gamma \underset{a}{\text{argmax}}\, Q(s_{t+1}^i, a)$ is referred to as the *target*. The algorithms then perform gradient descent on the loss $\mathcal{L}$ to learn an approximation of the Q-function. Note that we have written this using notation that assumes a single-agent operating in a Markov Decision Process for simplicity of notation.

In this work we use Twin-Delayed Deep Deterministic Policy Gradient (TD3) [32] a variant of Deep Deterministic Policy Gradient (DDPG) [24]. DDPG simultaneously learns a Q-function for estimating the values of states and a policy that selects actions that maximize the Q-function. Both policy μ and Q-function are learned simultaneously: the Q-function is learned by minimizing the Bellman error over a batch of data using gradient descent and the policy μ is learned by performing gradient ascent on the action component of the Q-function.

TD3 creates an empirically stabler version of DDPG by adding three simple tricks:

- The target is estimated using two Q-functions instead of one and taking the minimum of the two.
- The policy network is updated significantly less often than the value network.
- A small amount of noise is added to the action when estimating the value of the Q-function. This is based on the assumption that similar actions should have similar Q-values.

For more details, please refer to [32].

It is important to note that TD3 is a single-agent algorithm and that in multi-agent settings, there is additional instability induced by the changing policies of the other agents in the environment. Essentially, because the other agents in the environment have also changed, samples inside the buffer are *stale*, they no longer correctly represent either the rewards that would be received for taking an action in a given state nor is the subsequent state after taking that action the state that we would likely transition to. Algorithms that fail to address this issue and simply perform Q-learning are referred to as *Independent Learners*. This challenge can be addressed by algorithms like Multi-Agent DDPG [48] which use a Q-function that sees the observations and actions of all active agents.

In this work, we simply use *Independent Learners* with a shared policy: all of our agents use the same neural network to compute their actions. Surprisingly, we find this to be effective at designing performant policies despite the issue of *stale* buffer samples discussed above.

3.2.3 Car-Following Models

For our model of the driving dynamics, we use the default car-following model and lane-changing models in SUMO. We use SUMO 1.1.0 which has the Krauss car-following model [50]. The Krauss car-following model is quite simple: namely, the ego vehicle drives as fast as possible (subject to a maximum speed) while keeping a distance such that if the lead vehicle brakes as hard as possible, the ego vehicle is able to safely stop in time. For the parameters of

the model, we use the default values in the aforementioned SUMO version. The lane-changing model is also the default model described in [51].

3.2.4 Flow

We run our experiments in `Flow` [14], a library that provides an interface between the traffic microsimulators SUMO [13] and AIMSUN [52], and RLlib [53], a distributed reinforcement learning library. `Flow` enables users to create new traffic networks via a python interface, introduce autonomous controllers into the networks, and then train the controllers on many-CPU machines on the cloud via AWS EC2. To make it easier to reproduce our experiments or to try and improve on our results, our fork of `Flow`, scripts for running our experiments, reproducing our results, and tutorials can be found at https://github.com/eugenevinitsky/decentralized_bottlenecks.

3.2.5 Feedback Control and ALINEA

As our baseline for the performance of our RL controllers, we implement the traffic light controller from [41] (referred to here as ALINEA) and additionally design a decentralized variant of ALINEA that can be performed using AVs. The basic idea underlying ALINEA is to select an optimal bottleneck vehicle density and then perform feedback control around that optimal value using the ratio of red-time to green-time of the traffic light as the control parameter. We use the particular scheme outlined in [54] with some slight modifications.

Instead of operating around density, we feedback around a desired number of vehicles in the bottleneck which we denote as n_{crit}, a hyperparameter that we will empirically determine for our network. We update the desired inflow q as

$$\tilde{q}_{k+1} = q_k + K(n_{\text{crit}} - \hat{n})$$
$$q_{k+1} = \max(q_{\min}, \min(\tilde{q}_{k+1}, q_{\max}))$$

where K is the gain of the proportional feedback controller, $\hat{n}$ is the average number of vehicles in the bottleneck over the last T seconds, and $q_{\min}$ and $q_{\max}$ are minima and maxima of q to prevent issues with wind-up. We set $T = 25$, $q_{\min} = 200$, $q_{\max} = 14400$, and perform hyperparameter searches over K, n_{crit}, q_0. This desired inflow is then converted into a red-green cycle time via

$$c_k = r + g = \frac{7200 * L}{q_k}$$

where r is the red time, g is a fixed green time, and L is the maximum number of lanes. In this work, we set g to 4 which we empirically determined to be the amount of time needed to let two vehicles pass into the bottleneck. We perform this feedback update every 30 seconds. Finally, we initialize each of the traffic lights to have a cycle that is offset from its neighbors by 2 seconds to prevent the traffic lights from being completely in sync. Further details are provided in the code.

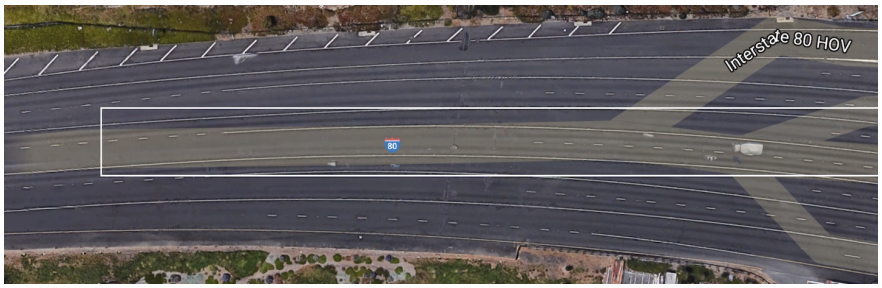


Figure 3.1: Bay bridge merge. The equivalent subsection that we study in this work is highlighted in a white square. Traffic travels from right to left.

Fortunately, we can apply the exact same strategy using autonomous vehicles instead of traffic lights where c_k is now the amount of time that an AV will wait at the bottleneck entrance before entering. However, the hyper-parameters will differ sharply as a function of the penetration rate. For a given penetration rate percentage, p, the expected length of the human platoon behind a given AV will be $\frac{1}{p} - 1$. Whereas in the traffic light case we can set arbitrary inflows, here every time an AV goes, $\frac{1}{p} - 1$ vehicles will follow it on average. As a consequence, to avoid congestion at lower penetration rates, the inflow needs to be a good deal lower. As we will discuss in Section 3.4, this leads to the decentralized control scheme under-performing traffic-light based control. For the exact hyper-parameters swept, see the appendix.

3.3 Experiments

3.3.1 Experiment Setup

We attempt to improve the outflow of the bottleneck depicted in Fig. 3.2, in which a long straight segment is followed by two zipper merges sending four lanes to two, and then another zipper merge sending two lanes to one. This is a simplified model of the post-ramp meter bottleneck on the Oakland-San Francisco Bay Bridge depicted in Fig. 3.1. Once congestion forms, as in Fig. 3.3, the congestion does not dissipate due to lower outflow than inflow and begins to extend upstream.

An important point to note is that in this work lane-changing is disabled for all the vehicles in this system. As we discuss in Sec. 3.4.4, this enables higher throughput but would require the imposition of new road rules at the bottleneck. Fortunately, this would only require painting some new lines that restrict lane-changing which should be relatively cheap.

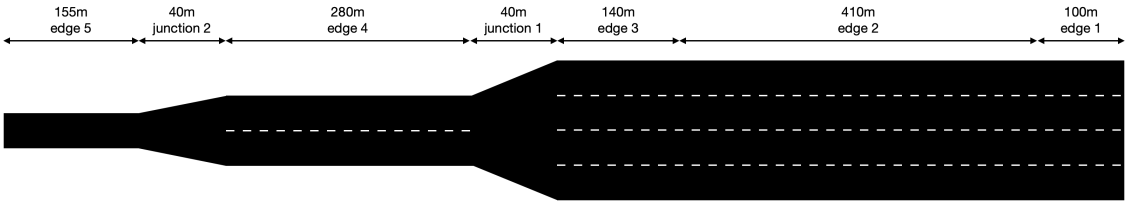


Figure 3.2: Long entering segment followed by two zipper merges, a long segment, and then another zipper merge. Scale is distorted to make visible relevant merge sections.

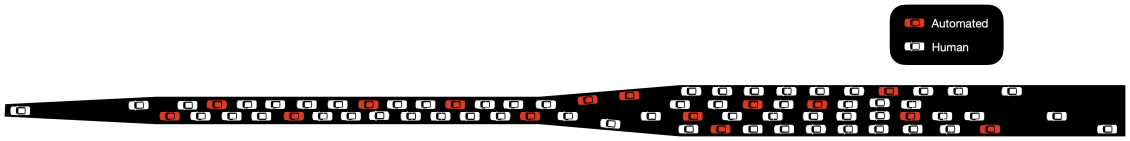


Figure 3.3: Congestion forming in the bottleneck. Congestion starts at the left (downstream) and propagates right (upstream). Red vehicles are automated, human drivers are in white.

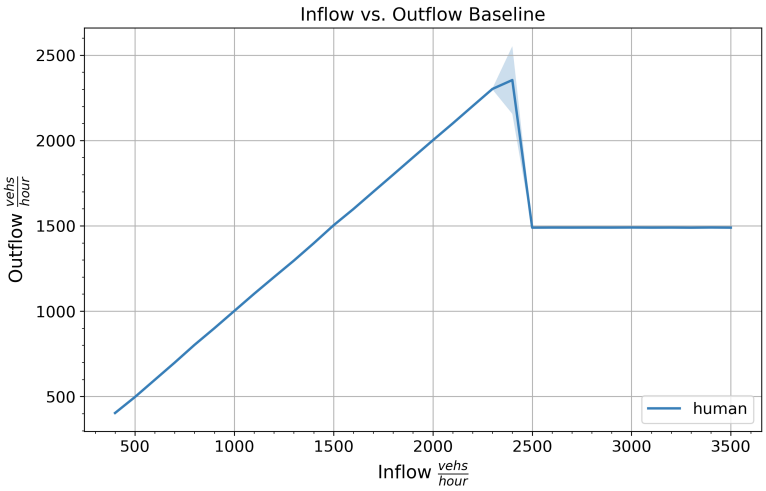


Figure 3.4: Inflow vs. outflow for the uncontrolled bottleneck. The solid line represents the average over 20 runs at each inflow value and the darker transparent section is one standard deviation from the mean.

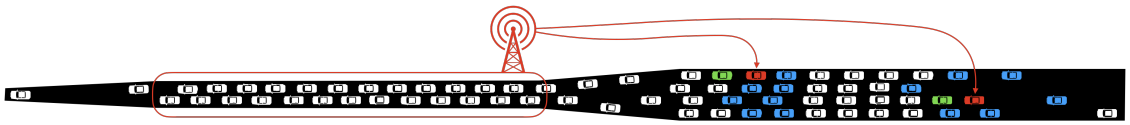


Figure 3.5: Diagram representing the different state spaces. The red vehicles can see the green vehicles in the minimal state space. In the radar state space, they can see both the blue and green vehicles. In the minimal and aggregate state spaces, the red vehicle also has access to information about the vehicle count in the bottleneck which we represent as a tower communicating information about the road segment enclosed in red. In addition to the states indicated here, the aggregate state space also contains the average speeds of edges 3, 4, and 5 (see Fig. 3.2).

3.3.2 Capacity Diagrams

Fig. 3.4 presents the inflow-outflow relationship of the uncontrolled bottleneck model. To compute this, we swept over inflows from 400 to 3500 vehicles per hour in steps of 100, ran 20 runs for each inflow value, and took the outflow as the average outflow over the last 500 seconds. Fig. 3.4 presents the average value and 1 std-deviation from the average across these 20 runs. Below an inflow of 2300 vehicles per hour congestion does not occur; above 2500 vehicles per hour congestion will form with high certainty. A key point is that once the congestion forms at these high inflows, at values upwards of 2500 vehicles per hour, it does not dissolve unless inflow is reduced for a long period of time. Identical inflow-outflow behavior is observed when lane-changing is enabled so a similar graph with lane-changing is omitted.

3.3.3 Partially Observed Markov Decision Process Structure

Here we outline the definition of the action space, observation function, reward function, and transition dynamics of the POMDP that is used in our controllers. We distinguish three cases that depend on the type of sensing infrastructure that will be available to the controller. The central concern is that the observations must give some way of identifying the state of the bottleneck (speed and density) to allow the AVs to intelligently regulate the inflow. Without some estimate of the bottleneck state, the AVs must be extremely conservative to ensure that the bottleneck does not enter congestion. The estimation can be done explicitly by acquiring the bottleneck state with loop detectors/overhead cameras or implicitly, by observing the behavior of vehicles around the bottleneck and inferring what the state of the bottleneck must be.

We have three possible state spaces that we investigate. The first state set we call the *radar state* as the states accessed would be readily available via an on-board radar and GPS. This is the state space that would be most easily implemented using existing technology on

an autonomous vehicle. In the radar environment, the state set is:

- The speed and headway of one vehicle ahead in each of the lanes and one vehicle behind. If the vehicle is on a segment with four lanes, it will see one vehicle ahead in each of the lanes and one vehicle behind in each of the lanes. A missing vehicle is indicated as two zeros in the appropriate position. Subject to some restrictions on sensing range, this information can be acquired via radar. Refer to Fig. 3.5 for a diagrammatic description. We also note that although AVs are only controlled on edge 3 (cf. Fig. 3.2), which is usually in a congested regime meaning leader and follower vehicles in all lanes are close to the ego vehicle, a current limitation of our work is that we default to assuming that our AVs has an infinite, unobstructed sensing range. However, we study the effect of restricting the sensing range in Sec. 3.4.4.

- The speed of the ego vehicle as well as its lane where lanes are numbered in increasing order from right to left.

- The edge number and position on the edge of the ego vehicle where the edge numbering is according to Fig. 3.2. This information would readily be available via GPS. We supplement this with an "absolute" position on the network, indicating how far the vehicle has traveled. This latter state is technically redundant and could be inferred from edge number and position.

- A counter that indicates how long the speed of the vehicle has been below 0.2 meters per second. This is used to endow the AV with a memory that allows it to track how long it has been stopped / waiting at the bottleneck entrance.

- A global time counter indicating how much time has passed.

Note that in the *radar state* information about the bottleneck is only indirectly available; it can only examine the states of visible vehicles and use it to infer information about the bottleneck state.

The second set of states would be available given appropriate loop sensing infrastructure or a sufficient number of overhead cameras distributed throughout the bottleneck. These states endow the AV with macroscopic information about the bottleneck. We refer to this as the *aggregate state*. Here the additional states are:

- The average speed of the vehicles on edges 3, 4, and 5.

- The number of vehicles in the bottleneck.

- A global time counter indicating how much time has passed.

Finally, the final set of states we consider is a significantly pruned state set in which we have hand-picked what we suspect to be a minimal set of states with which the task can be accomplished (which still includes information about the bottleneck state). We refer to this

as the *minimal state*. This state space should yield the fastest learning due to its small size. Here the states are:

- Total distance travelled.
- The number of vehicles in the bottleneck.
- A counter that indicates how long the speed of the vehicle has been below 0.2 meters per second.
- Ego speed.
- Leader speed.
- Headway.
- The amount of time our feedback controller described in Sec. 3.2.5 would wait before entering the bottleneck. This state is intended to ease the learning process since initially the vehicles can simply learn to imitate this value.

From these three potential sets of states, we form three combined state spaces that we study: radar + aggregate, minimal + aggregate, and minimal alone. Each of these represents a different set of assumptions on what sensing technology will be available, as is illustrated in Fig. 3.5. We characterize the relative performance of these different state spaces in Sec. 3.4.1. We note that each of these state spaces has some macroscopic information about the state of the bottleneck, which we found critical for getting good performance; for this reason, we do not include state spaces that contain no information about the bottleneck, such as radar alone or minimal without bottleneck counts.

The action space is simply a 1-dimensional acceleration. The vehicles enter the network at 25 meters per second and roughly maintain that speed as they travel along. Since our intent is for the controllers to stop at edge three and determine the optimal time to enter the bottleneck, we want to increase the likelihood of them coming to a stop on edge three. To increase the likelihood of sharp decelerations, we bound our action space between $\frac{1}{8}[-4.5, 2.6]$ and re-scale the actions by multiplying them by eight. The neural network weight initialization scheme we use (see appendix for details) tends to output actions bounded between $[-1, 1]$ at initialization time; this re-scaling scheme makes it likelier that large decelerations are applied.

While we could include lane-changes as a possible action, we leave this to future work, as it introduces a lot of complexity, is not guaranteed to yield better results, and causes the training to take longer. To prevent the vehicles from forming unusual patterns at the entrance of the bottleneck, control is only applied on edge 3 (edges numbered according to Fig. 3.2). However, states and rewards are received at every time-step and consequently actions are computed at each time-step: we simply ignore the controller output unless we are on the third edge.

We are trying to optimize throughput, so as our reward function we simply use the number of vehicles that have exited in the previous time-step as a reward

$$r_t(s_t, a_t) = n_t/N$$

where n_t is the number of vehicles that have exited in that time-step and N is a normalization term that was use to keep the cumulative reward at reasonable values. We use $N = 50$ in this work. Since the outflow is exactly the quantity we are trying to optimize, optimizing our global reward function should result in the desired improvement. This is a global reward function that is shared by every agent in the network.

However, we note a few challenges that make this a difficult reward function optimize. First, the reward is global which causes difficulties in credit assignment. Namely, it is not clear which vehicle's action contributed to the reward at any given time-step. Secondly, there is a large gap between when an action is taken and when the reward is received for that action. That is, a vehicle choosing to enter the bottleneck does not receive any reward directly attributable to that decision for upwards of 20 steps. Finally, the bottleneck being fully congested is likely a local minimum that is hard to escape. Once congestion has onset, it cannot be removed without a temporary period where the inflow into the bottleneck is reduced. However, a single vehicle choosing to not enter the bottleneck would have negligible effect on the inflow, making it difficult for vehicles to learn that decongestion is even possible.

3.3.4 Divergence Between Nash Equilibrium and Social Optimum

Here we provide a simple illustrative example of how, despite having a single, global reward function, open networks can lead to non-cooperative behavior. In the case where every vehicle receives the same reward at every time-step, it is simple to see that the Nash Equilibrium will be the same as the social optimum. However, vehicles sharing the same reward function but optimizing over different horizons can cause a divergence between the two equilibria. The key intuition is that although all of the vehicles are trying to optimize the same quantity, they only receive rewards while they are in the system as their trajectory terminates once they go through the exit. This creates a perverse incentive to remain in the system for longer than is socially desirable, leading to a divergence from the social optimum. Consider the following simplified, single-step variant of the bottleneck in which there are simply two vehicles. The problem has the following reward structure before we introduce the open-endedness:

- If both vehicles go, congestion occurs and they receive a reward of 1.
- If one vehicle goes and the other doesn't, no congestion occurs. Both vehicles receive a reward of 2.
- If both vehicles do not go, no outflow occurs and they receive a reward of 0.

This problem mimics the structure of the bottleneck where it is necessary to restrict the inflow to maximize the outflow. It is straightforward to see that the optimum is achieved

Table 3.1: One time-step model of the bottleneck reward structure. Here we make sure to reward the vehicle that exited the system, even though in an MDP its trajectory would have ended and no reward would have been given.

		Vehicle 2: Go	Vehicle 2: No Go
Vehicle 1	Go	$(1, 1)$	$(2, 2)$
	No Go	$(2, 2)$	$(0, 0)$

Table 3.2: One time-step model of the bottleneck reward structure where vehicles do not receive reward after they exit.

		Vehicle 2: Go	Vehicle 2: No Go
Vehicle 1	Go	$(1, 1)$	$(0, 2)$
	No Go	$(2, 0)$	$(0, 0)$

when one vehicle goes and the other doesn't and that this is both the social optimum and a Nash equilibrium. The game is depicted in Table 3.1 where it can visually be verified that (No-Go, Go) and (Go, No-Go) are Nash Equilibria but (No Go, No Go) is not an equilibrium point.

Open networks modify the problem in that once a vehicle exits it ceases to receive any reward. Therefore, the vehicle that goes does not actually observe any outflow and will receive a reward of zero.

- If both vehicles go, congestion occurs and they receive a reward of 1.
- If one vehicle goes and the other doesn't, no congestion occurs. The vehicle that went receives a reward of 0 while the other one receives a reward of 2.
- If both vehicles do not go, they receive a reward of 0.

In this setting, depicted in Table. 3.2, (No Go, No Go) is now a weak Nash Equilibrium. From the perspective of learning, this is a ubiquitous equilibrium as the vehicles that chose not to go will tend to accumulate a lot of reward as long as at least some other vehicles are going. An easy solution to remove this equilibrium is to adopt the perspective of the game in Table 3.1 and continue to reward vehicles even after they leave the system. However,

this would create a very noisy reward function as agents that exit the system earlier would receive a lot of reward from states and actions that they did not particularly influence. An alternative variant is to keep all the agents persistently in the system: run the system for some warm-up time to accumulate a starting number of vehicles and after that, any vehicle that exits is rerouted back to the entrance. We adopt this choice and reroute the vehicles during the training. However, this could lead to vehicles manipulating the bottleneck across reroutes and so we turn this rerouting behavior off when testing the policies after training.

3.3.5 Experiment Details

For the training parameters for TD3, we primarily used the default parameters set in RLlib [55] version 0.8.0, a distributed deep RL library. The percentage of autonomous vehicles varies among 5%, 10%, 20%, and 40%. During each training rollout, we keep a fixed inflow of 2400 vehicles per hour over the whole horizon. At each time-step, a random number of vehicles are emitted from the start edge. Thus, the number of vehicles in each platoon behind the AVs will be of variable length and it is possible that at any time-step any given lane may have zero AVs in it. To populate the simulation fully with vehicles, we allow the experiment to run uncontrolled for 300 seconds as a warm-up. After that, we run an RL rollout for 1000 seconds.

We use the traffic micro-simulator SUMO [13] for running our simulations. At training time, we use the re-routing technique discussed in Sec. 3.3.4 where vehicles are simply placed back at the beginning of the network after exiting. It is essential to note that for the multi-agent experiments we used a *shared controller*, all of the agents operate in a decentralized fashion but share the same controller.

For more details, see the Appendix.

3.4 Results

In this section, we attempt to provide experimental results that answer the following questions:

1. How does the ability to improve bottleneck throughput scale with available sensing infrastructure? With penetration rate?

2. Is there a single controller that will work effectively across all penetration rates?

3. Can we construct an effective controller that uses purely local observations?

3.4.1 Effect of Sensing

Here we compare the relative performance of the different sensing options across different penetration rates. Fig. 3.6 compares the evolution of the inflow-outflow curve of the three state spaces to the uncontrolled case (labeled human), the traffic light baseline (labeled ALINEA),

Table 3.3: Average outflow and its variance at an inflow of 3500 vehicles per hour, as a function of the penetration and the state space.

	minimal	**minimal + aggregate**	**radar + aggregate**
5%	1803 ± 83	1813 ± 114	1817 ± 80
10%	1829 ± 46	1863 ± 76	1888 ± 47
20%	1811 ± 29	1897 ± 46	1980 ± 48
40%	1878 ± 40	1910 ± 46	2034 ± 45

and the hand-designed feedback controller operating at a 40% penetration rate. Each of the state spaces outperforms the hand-designed feedback controller at every penetration rate and provides a 15% improvement in the outflow even at the 5% penetration rate. To study the evolution of the outflow with penetration rate, Fig. 3.7 illustrates the outflow at an inflow of 2400 as a function of penetration rate. Only the *radar + aggregate* state space is able to consistently take advantage of increasing penetration rate. Excitingly, its performance at a 40% penetration equals the performance of a traffic light-based controller. Table 3.3 summarizes the values of each of the different state spaces at an inflow of 3500 vehicles per hour.

3.4.2 Is There a Universal Controller?

In Sec. 3.4.1 we train a separate controller for each penetration rate. Atop the additional computational expense needed to train a new controller for each data-point, having one controller per penetration rate might require an accurate online estimation of current penetration rates so as to switch to the appropriate control scheme. Here we point out (see Fig. 3.8) that at least for the controllers studied here, this concern is justified: a controller trained at one penetration rate and evaluated at another will underperform a controller trained at the latter penetration rate. We also investigate a simple dynamics randomization strategy where we randomly sample a new penetration rate at each rollout and confirm that this can yield a controller that performs effectively across penetration rates albeit with some small loss of performance.

The key challenge is that the appropriate amount of time needed to wait before entering the bottleneck is a function of the penetration rate. As a simplified model to generate intuition, imagine that the bottleneck deterministically congests if more than 11 vehicles enter it. We will refer to an AV with N vehicles behind it as a *platoon of length N*. At a penetration rate of 10%, the average platoon length is 9. If we have two AV platoons ready to enter the bottleneck, one of the platoons must wait until the other platoon is almost completely into the bottleneck or else it will congest. At a 20% penetration rate (a platoon of average length

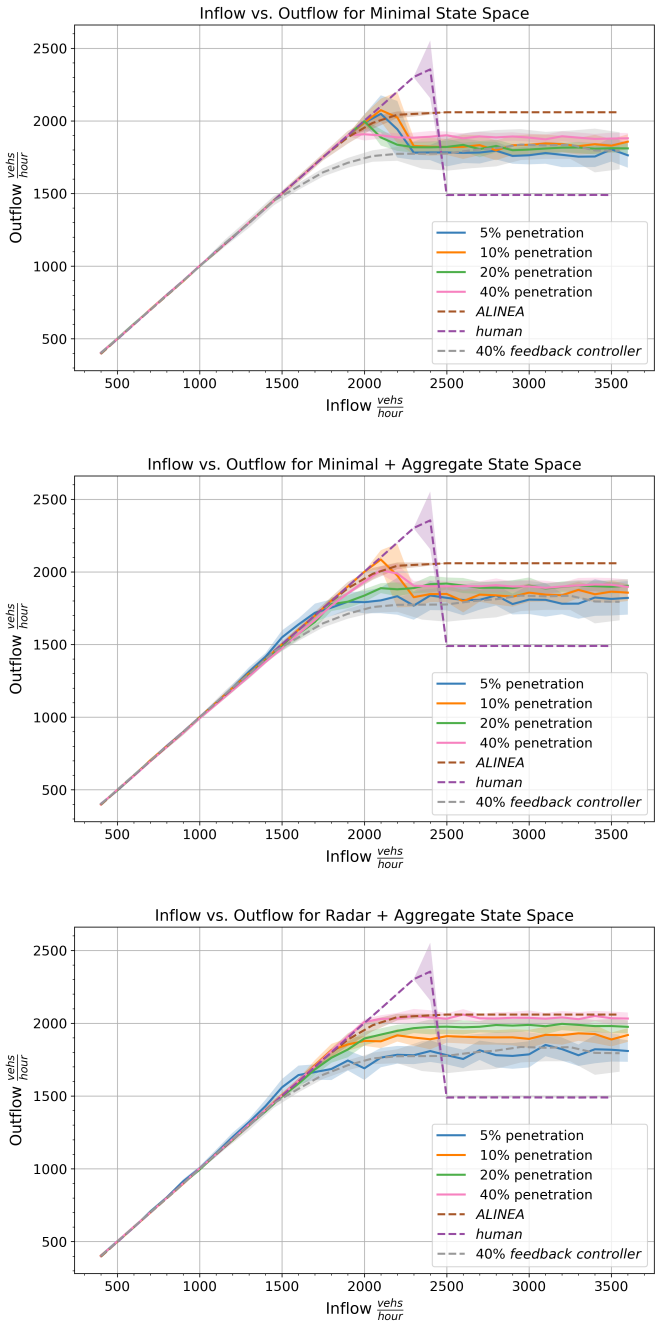


Figure 3.6: Inflow vs. Outflow curves for different penetration rates, for minimal, minimal + aggregate, and radar + aggregate (top to bottom). We plot performance of our controllers trained at four different penetration rates, traffic light baseline ALINEA, our feedback controllers at 40% penetration rate, and the uncontrolled case marked "human".

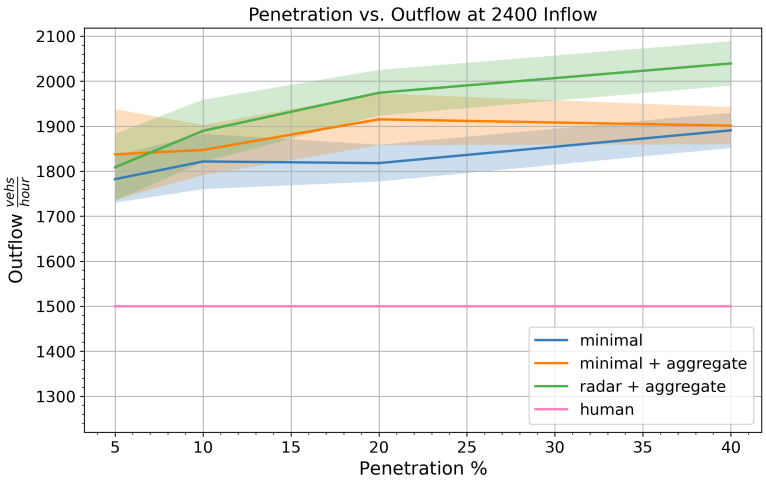


Figure 3.7: Evolution of the outflow as a function of the penetration rate for the three state spaces we are using. We also plot the uncontrolled human baseline for reference.

4), however, two platoons can go at once without worrying about running into congestion.

As a result, a controller trained at low penetration rates may be too conservative when deployed at higher penetration rates while a controller trained at high penetration rates may not be conservative enough at low penetration rates. As demonstrated in Fig. 3.8, a controller trained at a 10% penetration rate does significantly worse when deployed at a 40% penetration rate. Thus, if we use the controllers trained in Sec. 3.4.1, it will be necessary to use either infrastructure or historical data to identify the current penetration rate and deploy the appropriate controller. This motivates our attempt to find a single controller that is stable across penetration rates. We demonstrate that in return for some small degradation in performance, we can construct a single controller that performs robustly across penetration rates. To achieve this, we use *dynamics randomization* and for each trajectory we sample a new penetration rate p uniformly from $p \sim U(0.05, 0.4)$. We refer to these as *universal controllers* and the controllers trained at individual penetration rates in Sec. 3.4.1 as *independent* controllers. Fig. 3.9 shows the performance of the universal controllers compared to the controllers trained at a penetration rate and evaluated at the same penetration rate for each of the three state spaces we are using.

From Fig. 3.9, we can see that the results do not have a consistent trend as the universal controllers both over-perform and underperform at different penetration rates. For example, the minimal + aggregate controller gives up 100 vehicles per hour at a 5% penetration rate but outperforms the independent controller by 250 vehicles per hour at high penetration rates. However, we note that even the worst outcome at low penetration rates, the universal minimal controller, outperforms the uncontrolled baseline of 1550 vehicles per hour. Additionally, the universal radar + aggregate controller consistently provides an outflow of 1850 vehicles per

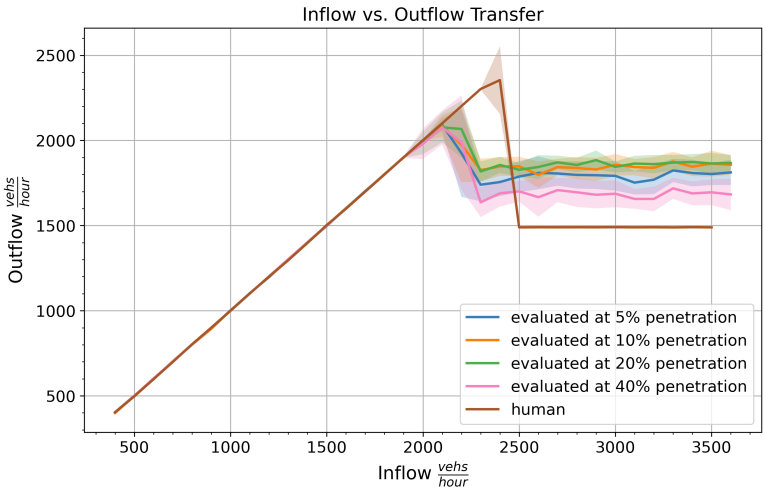


Figure 3.8: The inflow vs. outflow curve for a controller trained at 10% penetration on the minimal + aggregate state space, and evaluated at 5%, 10%, 20%, and 40% penetration. We also plot the uncontrolled human baseline for reference.

hour which achieves the desired goal of an effective controller independent of penetration rate.

3.4.3 Controllers Without Macroscopic Observations

The three state spaces studied earlier, minimal, minimal + aggregate, radar + aggregate, all contain the number of vehicles in the bottleneck (edge 4 in Fig. 3.2) in the state space as well as average speed data on edges 3, 4, 5 in the aggregate cases. Acquiring this information consistently (rather than through a lucky LIDAR or radar bounce picking up many vehicles ahead of the ego vehicle) would require either camera or loop sensing infrastructure. We would like to understand whether efficient control can be done without access to the number of vehicles in the bottleneck, a quantity we refer to as *congest number*. This would enable us to perform control that is fully decentralized in both action and observation, allowing us to deploy these systems with no additional infrastructure cost.

We attempt to answer this question by using the radar state space alone without any aggregate information. Thus, the vehicle only has access to the speeds and distances of the vehicles directly ahead of and behind it in each lane. Although it is not obvious how the controller will accomplish inference of the *congest number*, a few possible options:

1. There exists a scheme that does not actually depend on the number of vehicles in the bottleneck.

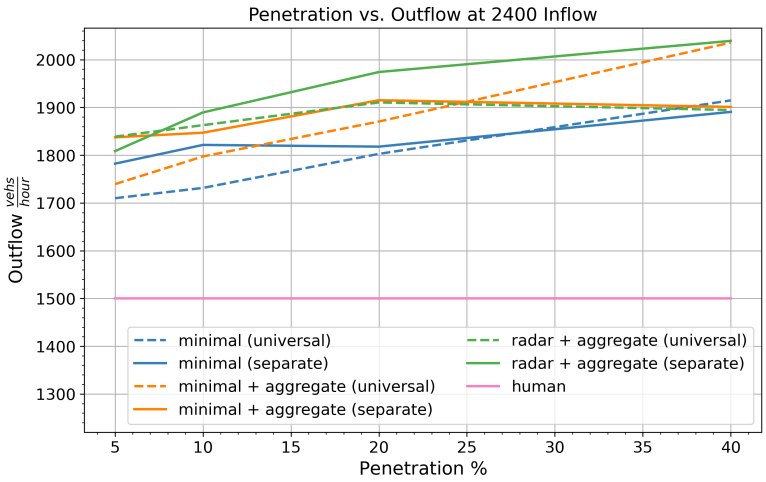


Figure 3.9: Evolution of the outflow as a function of the penetration rate for the three state spaces we are using. For each state space, we compare the universal controller trained using dynamics randomization and evaluated at different penetration rates, to the four independent controllers trained and evaluated at their own penetrations rate of respectively 5%, 10%, 20% and 40%. We also plot the uncontrolled human baseline for reference.

2. The number of vehicles in the bottleneck can be inferred from the distance to the visible vehicles or the speed of the visible vehicles. For example, if a vehicle observed in the bottleneck is stopped that likely indicates congestion while high velocities would indicate free flow.

3. Vehicles can learn to communicate through motion by adopting vehicle spacing and velocities that observing vehicles can use to infer the *congest number*. For example, a velocity between 2 and 3 m/s could indicate 0-5 vehicles in the bottleneck, between 3 and 4 m/s could indicate 6-10 vehicles, and so on.

While we are not able to conclusively establish which of the above hypotheses are active, Fig. 3.10 demonstrates the effect of removing any macroscopic information from the state space. The radar state space with no macroscopic data, marked in orange, deviates from the aggregate state space by about 100 vehicles per hour but still sharply outperforms an uncontrolled baseline (marked human). Since radar sensors are now standard features on level 2 vehicles, this suggests that a fully decentralized controller can be deployed using available technology.

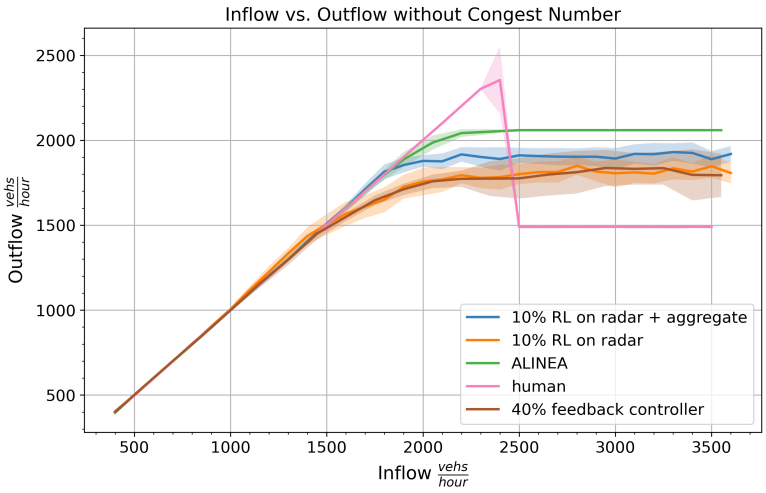


Figure 3.10: Evolution of the inflow vs. outflow curve for controllers trained at a penetration rate of 10%. We compare a controller trained on the full radar + aggregate state space to a controller only trained on the radar state space, which means it doesn't have access to the number of vehicles in the bottleneck. We also plot the performance of the traffic light baseline ALINEA, of our feedback controller at 40%, and of the uncontrolled curve marked "human".

3.4.4 Robustness of Simplifications

In this section, we attempt to relax some of the simplifying assumptions made in the design of our problem. Specifically, we investigate the effects of the following changes:

- Lane-changing. In the prior results, we have disabled lane-changing. We now study whether our best controllers are able to handle adding lane-changing to the human driver dynamics without retraining the RL controller.
- Simplifications in the "radar" model. Our "radar" state space does not take into account occlusions and distance limitations of radar and can return a vehicle an arbitrary distance away.

In Fig. 3.11, we enable lane-changing and examine how effective our controllers are as the penetration rate evolves. Unsurprisingly, at low penetration rates, there is a sharp reduction in outflow relative to the lane-changing disabled setting. The challenge is that when one of the AVs goes, other vehicles will rapidly lane-change into its lane which prevents the AV from restricting the inflow. As the penetration rate increases, when an AV at the front of the queue goes, a new AV rapidly arrives to replace it which consequently minimizes the impact of lane-changing. However, we note that when vehicles lane-change in SUMO, they instantly change lanes which may enable more aggressive lane-changing than is physically possible.

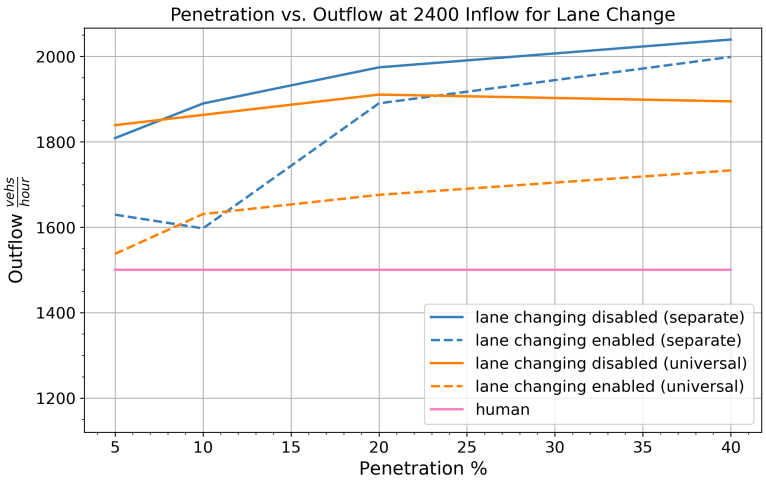


Figure 3.11: Evolution of the outflow as a function of the penetration rate for controllers trained on the radar + aggregate state space. We compare controllers that have been trained with lane-changing disabled, to those sames controllers when lane-changing is enabled at evaluation time. We compare both controllers trained on a fixed penetration rate of 5%, 10%, 20% or 40%, referred to as "separate", and controllers trained at a random penetration rate between 5% and 40% as explained in 3.4.2, referred to as "universal". We also plot the uncontrolled human baseline for reference.

Hence, the degradation in outflow might be lower in reality than it is in our simulator. Furthermore, disabling lane-changing at a bottleneck by painting new road lines should be relatively cheap.

As for the question of a more "realistic" radar model, Fig. 3.12 presents our attempt to restrict the range beyond which vehicles cease to become visible. We take a trained policy and cap how far it is allowed to see as an approximation of a restricted radar range. We try two restrictions, not seeing past 20 meters and not seeing past 140 meters. If there is no vehicle within that range, we set a default state with a distance of 20 or 140 meters respectively, a speed of 5 meters per second, and treat it as a human vehicle by passing a zero to the boolean that indicates whether a vehicle is human or autonomous. Other replacements in the state for vehicles that are too far away to see are possible, we could instead replace the missing states with a vector of -1's but found empirically that the replacement discussed led to better performance. We find that the *universal* controllers are relatively robust to this replacement which suggests that their controller is more independent of actual vehicle sensing and more dependent on macroscopic states. This ablation, simply ignoring vehicles that are too far away, is an extremely approximate model of how radar might work, and replacing it with more accurate radar models is a topic for future work.

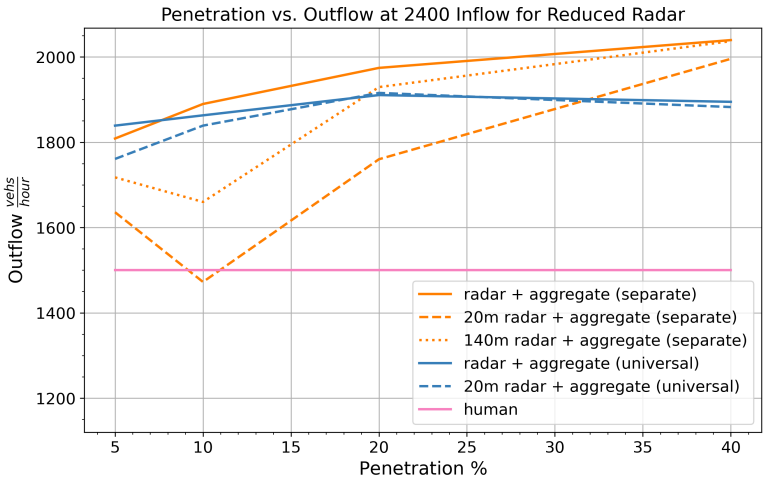


Figure 3.12: Evolution of the outflow as a function of the penetration rate for controllers trained on the radar + aggregate state space. We compare controllers that have been trained with the (normal) entire radar state space, to those same controllers when the radar is restricted to seeing only vehicles up to a distance of 20 meters or 140 meters at evaluation time. We compare both controllers trained on a fixed penetration rate of 5%, 10%, 20% or 40%, referred to as "separate", and controllers trained at a random penetration rate between 5% and 40% as explained in 3.4.2, referred to as "universal". We also plot the uncontrolled human baseline for reference.

3.4.5 Instability of Reward Curve

For purposes of reproducibility, we provide a few representative reward curves from some of the training runs. These should help establish a sense of what the expected reward is as well as provide a calibrated sense of what fraction of training runs are expected to succeed. Since we use 36 CPU machines and each training run requires 1 CPU, we are able to train 35 random seeds in parallel (1 CPU is used to manage all the trainings) and we keep the best performing one. The following figure presents the results of 9 of these random seed trials (for better visibility) from one of the training runs.

Fig. 3.13 and Fig. 3.14 represent reward curves from low penetration rate runs (10%) and high penetration rate runs (40%) respectively. A high-scoring run corresponds to an average agent reward of around 10, a reward slightly above eight corresponds to all the vehicles just zooming into the bottleneck without pausing, and a reward below 2 corresponds to the vehicles mostly coming to a full stop. As is clear from the figures, at low penetration rates the training is relatively stable and converges quickly. However, as the penetration rate increases the reward curves become extremely unstable, with rapid oscillations in the expected reward. This instability is not due to variations in the outflow as the std. deviation of the outflow is low but is likely the outcome of applying independent Q-learning in a multi-agent



Figure 3.13: A sample of runs from a seed sweep at a penetration rate of 10%. Most of the seeds converge to a good policy.



Figure 3.14: A sample of runs from a seed sweep at a penetration rate of 40%. There is significant instability that is expected from using single-agent algorithms in a multi-agent problem.

system, leading to non-stationarity in the environment. Methods that explicitly handle this non-stationarity by using a centralized critic such as MADDPG [48] may help reduce the instability in the training.

3.5 Conclusions and Future work

In this work, we demonstrated that low levels of autonomous penetration, in this case 5%, are sufficient to learn an effective flow regulation strategy for a severe bottleneck. We demonstrate that even at low autonomous vehicle penetration rates, the controller is able to improve the outflow by 300 vehicles per hour. Furthermore, we are able to use additional availability of AVs and at a 40% penetration rate get equal performance to actually installing a new traffic light to regulate the inflow.

However, many open questions remain. In this work, we use an independent Q-learning algorithm which leads to serious instability in the training. As discussed in Sec. 3.4.5, at high penetration rates many of the training runs become unstable, which makes it unclear if we are near to the optimal policy for those penetration rates. This is a challenging multi-agent RL task and it would be interesting to see whether multi-agent RL algorithms that use a centralized critic like MADDPG [48] and QMIX [56] would lead to more stable training. Furthermore, the training procedure takes 24 hours so finding algorithms that can perform with higher sample efficiency is critical.

One possible direction to pursue in future work is to increase the level of realism, adding both lane-changing and an accurate radar model that correctly accounts for obscurity. In preliminary investigations in Sec. 3.4.4, we found that lane-changing degrades the performance of our controllers as vehicles simply lane-change into the lane that is currently moving and thus avoid inflow restriction. It may be the case that this behavior can be avoided by more complex coordination between the AVs in which they explicitly arrange themselves to block this lane-changing behavior. Preliminary experiments we ran in which training was done with lane-changing enabled did not yield particularly strong results but this may be an artifact of our choice of the training algorithm.

Another open question is to investigate the effects of coordination between the vehicles. In the decentralized case, we still do not know the extent to which the AVs are coordinating in their choice of action. While implicit coordination is possible due to vehicles being aware of which nearby vehicles are also autonomous, we have only provided circumstantial evidence that this is actually occurring. In the case of lane-changing, such coordination may be needed to prevent human drivers from skipping lanes and decreasing the total outflow. Additionally, we do not use memory in any of our models which may be limiting the effectiveness of our controllers. Using memory-based networks such as LSTMs could be an interesting direction for future work.

One approach we intend to explore to explicitly enable coordination is to allow the AVs to communicate amongst themselves. Perhaps by broadcasting signals to nearby vehicles, the AVs can learn to coordinate platoons in such a way that changing lanes no longer appears

advantageous to the human driver and they will remain in their lane. Furthermore, if communication proves useful, it is possible that the AVs may develop a "language" that they use for coordination. Examining whether such a "language" emerges is a future thread of work.

Finally, an exciting potential consequence of these results, given that they're decentralized and only use local information available via vision, is that human drivers could potentially implement these behaviors. Investigating whether this scheme could be deployed via human driving, whether by constructing a mobile app that provides instructions or by teaching a new driving behavior, is a direction we hope to explore in the future.

While this work has demonstrated the potential of multi-agent reinforcement learning for bottleneck control in a microscopic traffic microsimulator, several challenges remain for broader application. The training instability at high penetration rates and the computational requirements of these approaches highlight the need for more efficient simulation environments that can support large-scale multi-agent learning while maintaining realistic traffic dynamics. These limitations motivate the development of new simulation frameworks that can bridge the gap between fast computation and detailed modeling of complex driving behaviors, which forms the foundation for the work presented in the next chapter.

Chapter 4

Nocturne: Multi-Agent Driving Benchmark

The multi-agent reinforcement learning challenges identified in the previous chapter highlight a fundamental limitation in developing scalable traffic control algorithms: the need for simulation environments that can support both realistic driving dynamics and fast computation required for large-scale learning. Current traffic microsimulators like SUMO, while detailed, operate too slowly for the billions of environment interactions that can be needed to train effective multi-agent policies. This chapter introduces Nocturne, a fast driving simulator designed to bridge this gap by enabling fast multi-agent learning on driving scenarios derived directly from trajectory data. Rather than modeling traffic flow abstractly, Nocturne reconstructs complex driving scenarios from real-world data while maintaining computational efficiency. It further introduces human-inspired partial observability, with the goal of learning accurate human driving modeling.

This chapter is based on [57]: Eugene Vinitsky*, Nathan Lichtlé*, Xiaomeng Yang*, Brandon Amos, and Jakob Foerster. "Nocturne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world." *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

4.1 Introduction

The bottleneck control experiments presented in Chapter 3 demonstrated the potential of multi-agent reinforcement learning for traffic control, but also revealed significant computational challenges. Training effective multi-agent policies required substantial computing resources and time, largely due to the simulation speed limitations of traditional traffic microsimulators. These limitations become particularly pronounced when scaling to more complex scenarios with larger numbers of interacting agents.

This chapter presents *Nocturne*, a new simulator and benchmark designed in part to address

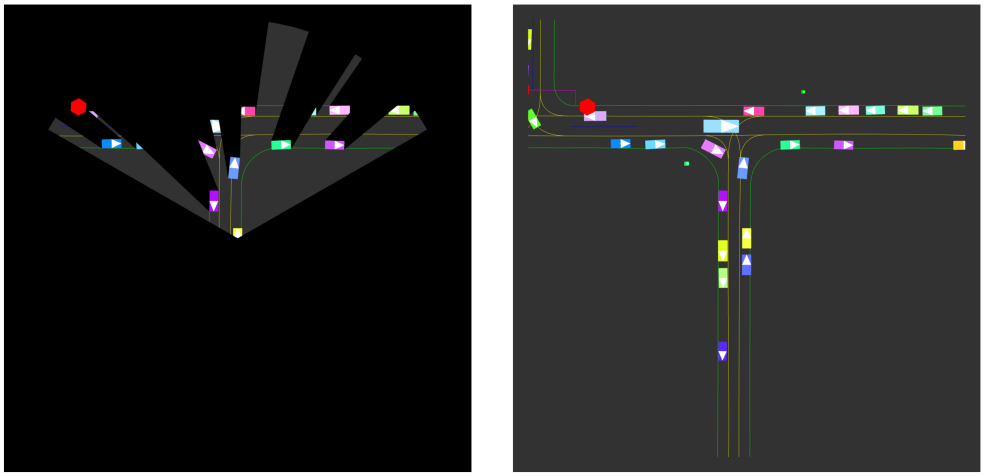


Figure 4.1: A visual depiction of the obstruction model used to represent the objects that are visible to the agents. (Left) Obstructed view with the viewing yellow agent in the center of the cone. (Right) Original scene centered on the cone agent with an unobstructed view.

these computational bottlenecks while enabling research into multi-agent coordination under realistic partial observability constraints. Instead of combining the challenges of coordination with computationally expensive feature extraction from images, Nocturne is a 2D simulator that generates vector representations of the set of objects and road points that would be visible to an idealized human driver. This approach provides the computational efficiency needed for large-scale multi-agent learning while maintaining the essential elements of realistic driving scenarios built directly from trajectory data.

Crucially, Nocturne is not just a simulator. Instead, it is built upon open-source driving data and features a diverse set of real-world scenes (see Fig. 4.2) that probe the ability of agents to safely navigate and coordinate in complex scenes such as intersections, roundabouts, parking lots, and highways. We use this data as a source of experts for imitation, to flexibly vary the number of controlled agents in the scene, and as a train-test split for validating the generalization ability of a human-driver model.

The challenge we propose is to learn (or otherwise design) policies that achieve the same set of final states as human experts (throughout this chapter we will call this the *goal rate* and the final state the *goal position*) while achieving a 0% collision rate. Achieving a high goal rate alongside a negligible collision rate is challenging for any policy design scheme (both learning and non-learning) due to the combination of the high-dimensional state space, the partial observability of the scene, the large number of interacting agents, and the decentralization of the policies at test time. The secondary challenge in Nocturne is to find policies that closely mimic human behavior at the trajectory level. Building on top of the human data we provide an evaluation scheme and off-the-shelf baseline implementations including evaluations.

We provide a guide to the construction and rules of the benchmark as well as results

on using Nocturne to design agents and test their capabilities and human-similarity. We demonstrate that learning effective agents in Nocturne is challenging; tuned RL and imitation learning baselines struggle to successfully complete the highly interactive scenes. Finally, we demonstrate that the agents achieve relatively low distance to the expert trajectories and show in Appendix B.2 that there does not appear to be a zero-shot coordination problem [58] at this level of agent capability.

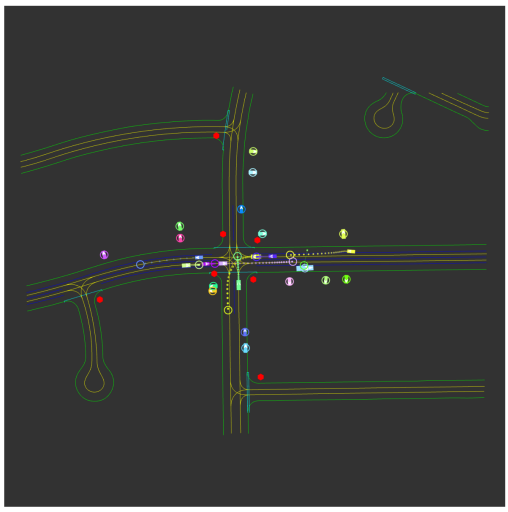
4.2 Related Work

Multi-agent traffic simulation tools and benchmarks. In terms of multi-agent driving benchmarks that require both acceleration and steering control, there are several closely related benchmarks that differ in terms of ease of implementing multi-agent interaction, being data-driven, 2D vs. 3D, the mechanism by which they support partial observability, or the rate at which they can construct the partially observed state for an agent. We summarize some of these differentiating features in Table 4.1.

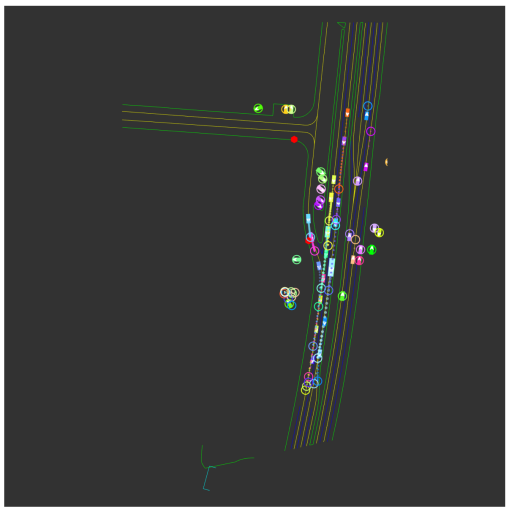
The closest works to ours are BARK [59], SMARTS [60], and MetaDrive [61]. SMARTS [60] supports multi-agent driving in a wide variety of interactive driving scenarios and offers a large set of default human driving behaviors to use for testing autonomous vehicles. BARK [59], like Nocturne, is a 2D goal-driven simulator with support for external datasets and contains a wide variety of large-scale scenarios and multi-agent support. Finally, MetaDrive [61] also supports real-world data and multi-agent interaction and is able to achieve 300 FPS image rendering by rendering lower fidelity images. The main differentiating feature from these works is the support for acquiring the set of objects that are visible without requiring the rendering of camera images; to our knowledge Nocturne is the only available simulator that can jointly compute an agent's visible objects and the agent's dynamics step above 2000 steps-per-second (see Appendix B.6 for exact details). Additional simulators are summarized in Table 4.1.

Partially observed multi-agent benchmarks. There are a wide variety of partially-observable multi-agent benchmarks not focused on driving. Within the card-playing domain, Hanabi [72] is frequently used to investigate coordination under partial observability and features between 2-5 agents. The Starcraft multi-agent benchmark (SMAC) [73], perhaps the most ubiquitous MARL benchmark, features many agents and high-dimensional observations but does not come with human data and many algorithms now achieve perfect performance in this challenge [74]. Melting Pot [75] features a huge diversity of many-agent mixed-motive challenges but does not have available expert data. Finally, games such as Poker, Stratego, and Diplomacy are frequently studied.

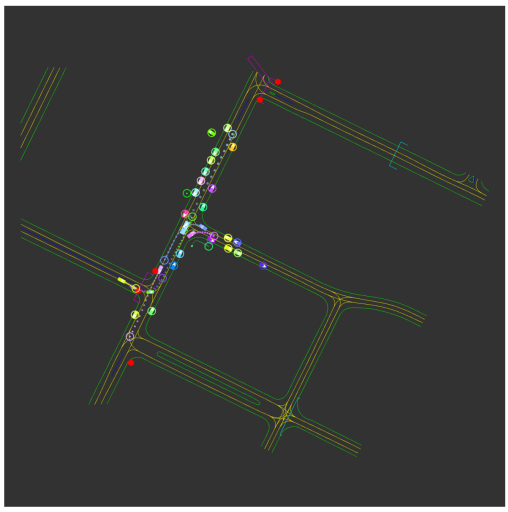
Open-Source Trajectory Data. We provide a brief overview of large driving datasets that contain both map information and annotated trajectory data for vehicles, pedestrians, and cyclists. Available trajectory data can be categorized along the size of the dataset, the



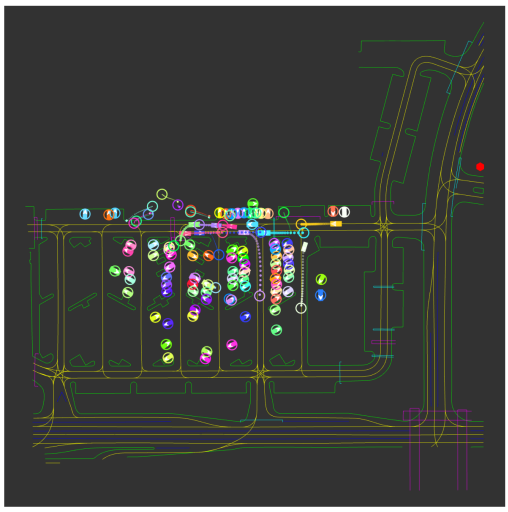
(a) A four-way stop.



(b) A straight road with merging vehicles.



(c) Unprotected turns with conflicting routes.



(d) A crowded parking lot.

Figure 4.2: Four scenes demonstrating the diversity of the navigable scenes in Nocturne. Colored circles represent the goal position of the corresponding colored agent. Dots represent the trajectory of the agent, with opacity increasing as time goes on. Videos of experts negotiating these scenes can be found at https://github.com/facebookresearch/nocturne.

Table 4.1: Comparison of representative driving simulators. State-based partial observability refers to whether the set of visible objects can be queried. Expert data refers to whether recorded trajectory data is available for the scenes. Baseline human models refers to whether explicit models of human driving are available to control road objects in the scene.

Simulator	Multi-agent Support	2D/3D	State-Based Partial Observability	Expert Data	Baseline Human Models
CARLA [62]		3D			✓
SUMMIT [63]	✓	3D			✓
MACAD [64]	✓	3D			✓
Highway-env [65]		2D			✓
Sim4CV [66]		3D			
Duckietown [67]		3D			
SMARTS [60]	✓	2D			✓
MADRaS [68]	✓	2D			✓
DriverGym [69]		2D		✓	✓
DeepDrive-Zero [70]	✓	2D			
MetaDrive [61]	✓	3D		✓	✓
VISTA [71]	✓	3D		✓	✓
Nocturne	✓	2D	✓	✓	

diversity of data in the dataset, the method of collection, and the available annotations of the scene (e.g. maps, traffic lights, road object trajectories). This work uses the Waymo Motion dataset [76] as it has a high diversity of scenes across many cities and contains annotations for cyclists, pedestrians, and traffic lights, features that will be used in future versions of Nocturne. However, we note that Argoverse 2 [77] and nuScenes [78] have similar features and could have been used; nuScenes also has complete traffic light annotations instead of annotations only for ego-observed traffic lights. The Lyft Level 5 dataset [79] appears to be the largest available dataset and has thorough road and vehicle annotations but is drawn from a single stretch of road. For non-egocentric datasets, the INTERACTION dataset [80] contains a diversity of scenes across both cities and countries, thereby capturing a variety of driving styles and norms, but is an order of magnitude smaller than other datasets.

4.3 Benchmark Construction

4.3.1 Defining a Nocturne Scene

In the following sections, we will refer to objects that can move (vehicles, pedestrians, cyclists) as *road objects* and anything that cannot move (lane lines, road edges, stop signs, etc.) as *road points*. *Road points* are connected together to form a polyline. We will refer to the type of road polyline that should not be crossed by vehicles as a *road edge*.

Nocturne scenes require a map consisting of polylines, a set of initial and final road object positions, and optionally a set of trajectories for the road objects. Nocturne currently acquires its scenes, goals, and expert trajectories from the Waymo Motion dataset [76] but can be easily configured to support any dataset that represents its road features as points or polylines. The Waymo Motion dataset consists of 487004 nine-second trajectory snippets discretized at a rate of 10 Hz with the first second intended to be used as context and the latter eight seconds to be used for prediction.

One challenge of selecting and constructing the scenarios for Nocturne is that these trajectories are collected by labeling the vehicles observed by a Waymo car as it drives. Consequently, we do not have a complete birds-eye view of the scene. Hence, there are cars that may have been in the scene that were not visible to the Waymo vehicle and therefore are not included in the dataset; for the same reason, the expert trajectories are also incomplete and may not persist throughout the entire duration of the rollout. In other words, the expert trajectories contain agents that unpredictably flicker in and out of existence. Similarly, the set of traffic lights that were visible to the Waymo vehicle may be insufficient to uniquely determine the underlying traffic light state. For this reason, this first version of the Nocturne benchmark comes with a few restrictions: we do not use traffic lights and we only include vehicles available at the first time-step of the scene to avoid collisions induced by unpredictable vehicle appearances. We filter out scenes that contain traffic lights which leaves the majority of the remaining scenes as roundabouts, unsignalized intersections, arterial roads, and parking lots. This leaves the benchmark consisting of 134453 snippets.

4.3.2 Partial Observability Model and Collision Handling

To support the investigation of coordination under partial observability, agents in Nocturne come with a configurable view-cone as is shown in Fig. 4.1. An element (road object or road point) is considered observable if there is a single ray in the cone that intersects with that element that does not pass through any road object on its path to the element. Stop signs are always visible if they are within the agent view-cone even if they would be obscured on the assumption that they would be raised at a sufficiently high level. We select the angle of the cone to be 120 degrees (approximately the range of binocular vision) and 80 meters radius. We note that this does not perfectly mimic a human visual model which has additional complexities such as dynamic variations in the functional field of view [81], phenomena relating to interactions between visible objects such as crowding [82], and the

necessary ability to pay attention to objects further than 80 meters away at high speeds. Furthermore, we currently represent our vehicles as rectangles and so this model of partial observability will be overly pessimal and neglect situations such as the ability for a driver to sometimes see over the hood of a car or 3-D effects like tall cars seeing over small ones.

The primary challenge in computing which road points and objects are observable is their relatively large number: a scene contains about 30 vehicles and 4800 road edge points on average. We use a Bounding Volume Hierarchy (BVH) to maintain the road objects and select candidates for potentially observed vehicles. We build the BVH using approximate agglomerative clustering [83] to generate a high-quality BVH. When computing the observed objects, we first use the BVH to select the candidates that lie in the *axis-aligned bounding box* (AABB) of the conic view field. Since there are a comparatively larger number of all types of road points (about 16000 on average), we use a 2D range tree [84] to maintain all of the road points. When computing the observed road points, we do a range search in the 2D range tree to select the candidates that lie in the AABB of the conic view field. For both vehicles and road points, once we have the candidates in the conic view-field, we perform a brute-force visibility check for the object and road points respectively by ray-casting from the viewing agent to all the candidate vertices. Finally, these data-structures are similarly used to accelerate collision checking.

4.3.3 Construction of the Partially Observable Stochastic Game

Definition of the state space

Nocturne supports two possible state representations: a rasterized image and a vectorized representation of that image. While we provide default state representations, we consider it fair game on the benchmark to use any other representation as long as only objects that are visible under the conditions in Sec. 4.3.2 are presented to the agent. Conforming to these consistent rules about visibility allows for a fair comparison between different algorithmic approaches.

For the default vectorized representation, we adopt a fully descriptive set of features (speed, angle, width, length, etc.) for the road objects and use the VectorNet [85] representation for the road points. We will refer to the vehicle whose observation is being returned as the *ego vehicle*. Note that by default all features that can be placed into relative coordinates are returned in relative coordinates to the ego vehicle (e.g. speed is relative speed, heading is relative heading, etc). The agent goal is set to be the final position, speed, and heading of the expert agent. An agent is considered to have achieved its goal if it is within 1 meter of the final position and within $1\,\frac{\text{m}}{\text{s}}$ of the final speed of the agent and 0.3 radians of the final heading when the target position tolerance is achieved. For exact observations for the ego object, road objects, and road points, as well as the padding mechanism used to handle variations in the number of observed objects / points / stop signs see Appendix B.1.

Action Space

Vehicles are driven by acceleration and steering commands that are passed to a bicycle model to update the vehicle state. In the experiments used in this chapter we use 6 discrete actions for acceleration, 21 discrete actions for steering, and 5 discrete actions for head tilt with the acceleration actions uniformly splitting $[-3, 2]$ $\frac{m}{s^2}$, the steering actions between $[-0.7, 0.7]$ radians, and the head tilt between $[-1.6, 1.6]$ radians. For more details on the vehicle model, see Appendix B.4.

Environment Dynamics and Goals

The total length of the expert data is 9 seconds, discretized into steps of size 0.1 seconds. For the first 1 second of the episode, all vehicles obey the expert policy. This is used to construct a history of observations for each agent that can be used to initialize or warm up the policy. After this transitory period, the episode continues for a fixed length of $T = 80$ steps. Agents are provided with a target position and target speed that are taken from the final speed and position of the expert trajectories. If an agent achieves their goal they receive a reward of T and are removed from the system. A vehicle will also be removed if it collides with any road edge or object.

In addition, there is a process for selecting the set of vehicles that are controlled in the environment. First, we only control vehicles that at some point have a speed above $0.05\frac{\text{m}}{\text{s}}$ and that are more than 0.2m from their goal; in general, these are vehicles that need to move to get to their goal. From this set of vehicles, we remove all vehicles that are already at their goal. Next, we randomly select up to a maximum of 20 of them and set the remainder to replay expert trajectories. This latter process for keeping a maximum number of controlled vehicles is used in this chapter due to constraints of the RL library and is not an aspect of the benchmark.

4.3.4 Rules of the Benchmark

We outline here a few rules that we expect solvers to respect to ensure consistency between solutions.

- The size of the view cone is fixed to 120 degrees and a distance of 80 meters. This ensures consistency between the level of partial-observability each controller must handle. Users can also tilt the head of the driver by up to 90 degrees in either direction to acquire more information.

- Only the first 1 second of the trajectory can be used as context or to warm-start a memory-based controller; control of the vehicles must start at 1 second into the trajectory.

- A trajectory that successfully reaches the goal is only considered valid if it reaches the goal within the 8 second time-window. All the scenes are easily solved without a time-constraint by simply creeping forwards slowly.
- The environment comes with default rewards and observations but any amount of reward-shaping or observation sharing at training time is valid. However, at test time only information that is directly observable to the agent can be used as input to the policy. For example, sharing information about other agents' goals would be valid at train time but not at test time.
- Adding additional map information to the agent state space beyond the information provided by default is valid. This corresponds to humans often having knowledge about map layout beyond what is within eyesight.
- The bounds on the action space should be respected: the acceleration should be bounded between $[-6, 6]\,\frac{\mathrm{m}}{\mathrm{s}^2}$, the change in heading should not be faster than 40 degrees per second, and driver head tilt should be maintained within 90 degrees in either direction. This rules out solutions that rapidly get to their goal by using accelerations that are outside of possible vehicle speed bounds or excessively sharp turns.

We note that these rules may make some of the scenes unsolvable. For example, real human drivers are not randomly initialized into a scene with a maximum of 1 second of prior context; this context may be critical to safely navigating the scene under the time constraints. Additionally, our model of human perception is not an exact match for true human perception which can extend well beyond 80 meters under certain circumstances. It is possible that there may be missing context at this viewing distance which is crucial for safe navigation. We view these constraints as similar to the challenge of label noise in supervised datasets; our constraints may place an upper limit on the percentage of goals that can be achieved but the benchmark still constitutes a valid comparison between methods as long as the constraints are respected.

4.3.5 Unusual features of the Benchmark

For completeness, we note a few oddities of the benchmark that we believe are critical for potential solution designers to be aware of. These challenges are often properties of labeling noise and the egocentric view under which the data was collected.

Filtered out pedestrians, cyclists, and traffic lights

While the dataset contains scenes with traffic lights, pedestrians, and cyclists, the first set of Nocturne environments, NocturneV1, operates solely on scenes that have been filtered to not include traffic lights. Additionally, when constructing the environment, we remove all pedestrians and cyclists from the scene. There is still an option to include them in the visible

state, which we do for behavioral cloning. Future versions of the benchmark will consider joint learning of pedestrians, cyclists, and vehicles but for NocturneV1 we only consider vehicles as appropriate modeling of pedestrian and cyclist dynamics is challenging.

Egocentric data collection

The Waymo Motion dataset that forms the basis of the first version of Nocturne is collected by driving a sensor-equipped car and recording the trajectories of all visible vehicles. Thus, in the original data vehicles may have trajectories that are shorter than the full 9 seconds and may only appear midway through the trajectory or appear and disappear throughout the trajectory when they are obscured from the view of the sensing car. Rather than suddenly teleport cars into the scene midway through an episode, we choose to only use cars that have valid states beginning of the episode. This reduces the total number of vehicles that might appear in the episode but does not change the feasibility of any of the agent goals.

Infeasible goals

Roughly 3% of the vehicles in the dataset have an expert trajectory that crosses an impassible road edge. This is due to labeling errors in the dataset where, for example, small gaps in road edges are occasionally missed that make the road edge crossable. To ensure a benchmark where all goals are achievable, we compute all trajectories where crossing a road edge was necessary to achieve the goal and set these vehicles to replay their expert trajectory rather than be controlled; this corresponds to 3% of all vehicles. Finally, 2% of vehicles in the dataset are initialized in a colliding state. This primarily occurs in parking lot scenes where a vehicle slightly overlaps with the boundary of a parking spot or a nearby vehicle. These vehicles are also removed. For the calculations of these statistics, see Appendix B.5.

Illogical goals

Occasionally, agents may have goals that seem illogical; for example, agents may be asked to come to a full stop in the middle of a highway. While these goals may seem odd, they are actual states that were achieved by humans driving on the roadways. These odd goals are often a consequence of unobserved objects such as drivers queued up that make the vehicles stop so as to avoid a collision that was not observed by the egocentric data collection process.

4.4 Experiments Setup

We run a reinforcement learning and an imitation learning baseline to demonstrate that these tasks do not appear to be easily solved even after billions of steps. We also briefly investigate whether the policies appear to have a zero-shot coordination challenge wherein policies perform well when paired with agents from their seed but are incompatible with policies from a different training run.

We test the following methods:

- APPO [86] trained in multi-agent mode with a shared policy i.e. every agent is controlled by the same policy but in a decentralized fashion.
- Behavior Cloning [87, 88].

For the RL method, in addition to the fixed-bonus for achieving the goal, we add the following dense reward to encourage the agent to make progress toward the target position, speed, and heading:

$$r_t = 0.2 \times \left(1 - \frac{||x_t - x_g||_2}{||x_0 - x_g||_2}\right) + 0.2 \times \left(1 - \frac{||v_t - v_g||_2}{40}\right) + 0.2 \times \left(1 - \frac{f(h_t, h_g)}{2\pi}\right) \quad (4.1)$$

where x_t is the position at time-step t, x_g is the goal position, v_t is the speed at time-step t and v_g is the target speed, h_t and h_g are the current and target heading in world coordinates (i.e. not in a relative frame). $f(h_t, h_g)$ returns the minimum angle between h_t and h_g. Since coordinates are in a relative frame and the agent cannot observe the world frame, note that $f(h_t, h_g)$ is an observation provided to the agent. The use of this dense reward, which certainly affects the form of the optimal policy, is not a necessary component of the benchmark and here is just used to generate good policies.

All experiments are evaluated on the same 200 randomly selected scenes from the validation set and statistics are computed over an 8 second rollout. The width in all the plots represents twice the standard error. For more details and hyperparameters for both methods and architectures, see Appendix B.1.

4.5 Results and Analysis

4.5.1 Success rate of baselines

Fig. 4.3 shows the performance of the agents on the training set after $3e9$ steps which takes approximately two days on 1 GPU and 10 CPUs. Each line corresponds to a policy trained on a fixed subset of the training scenes. We score our policies in two primary ways: the fraction of vehicles that achieve their goal (goal rate) and the fraction of vehicles that collide with another vehicle or road edge (collision rate). Note that this use of collision rate differs from its use in papers such as [89, 90] which compute collision rate as the fraction of scenes that contain a vehicle-vehicle collision.

We investigate the effect of the size of the dataset on the train and test performance. In the right half of Fig. 4.3 we can see that the inclusion of a larger training dataset decreases the performance of the algorithms up to 1000 files as they struggle with the diversity of the data. However, the inclusion of additional data narrows the magnitude of the train and test gap and closes it fully at 10000 files. However, it is still possible that a divergence between

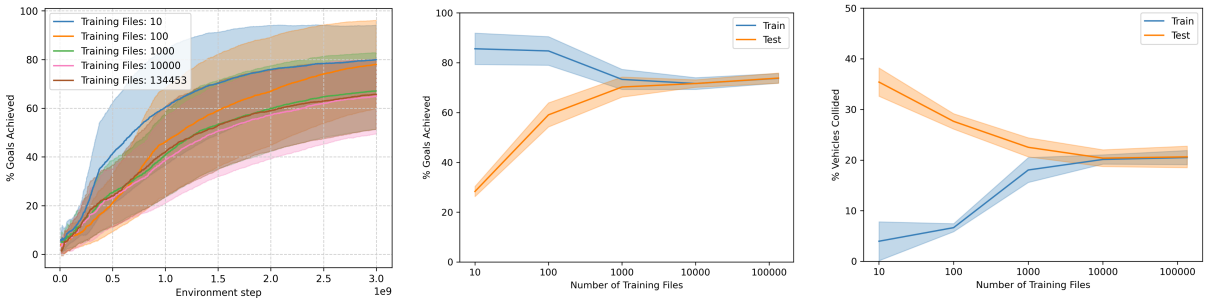


Figure 4.3: (Left) Success at getting to the specified goal on the training data as a function of number of environment steps. "Training Files: X" means the agent was trained on X fixed scenes sampled from the training dataset. (Middle) Percent of agents that achieved their goals. (Right) Percent of agents that collided.

train and test might re-emerge as agents become more capable on the train set and approach a 100% goal rate.

Finally, Table 4.2 compares the APPO and BC agents. For the APPO agent, we include the results for the agent trained on 10000 training files. The expert playback row refers to replay of the expert trajectories.

Table 4.2: Overview of metrics across methods for an 8 second rollout.

Algorithm	Collision Rate (%)	Goal Rate (%)	ADE (m)	FDE (m)
Expert Playback	4.9	100	0	0
APPO	20.3 ± 0.8	71.7 ± 0.7	3.1 ± 0.2	6.1 ± 0.3
BC	$38.2 \pm .1$	25.3 ± 0.1	5.6 ± 0.1	9.2 ± 0.1

4.5.2 Human-agent trajectory similarity

We analyze the results of our experiments with respect to how human-like the resultant policies are using displacement error between expert and agent trajectories as our metric (i.e. L2 distance between the agent and expert trajectories). To align with the definition used in other works [89, 90] we disable the removal of vehicles upon collision / reaching the goal. However, we note a few dissimilarities that make comparisons with other works difficult. First, agents are provided with a goal position, a feature that is often not available to other predictive methods. Second, our experts are stepped in scenes that may contain pedestrians or cyclists but our agents are replayed in the same scene without the corresponding pedestrians or cyclists. This can make the magnitude of the displacement error not directly comparable

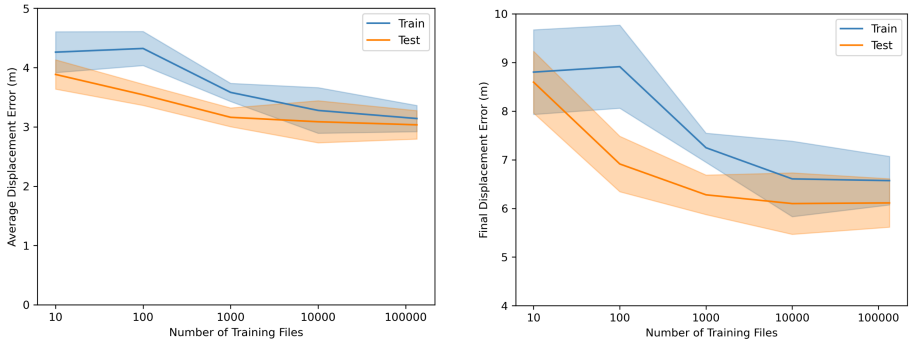


Figure 4.4: (Left) Average displacement error (mean l2-distance between an agent and an expert at each time-step). (Right) Final displacement error (l2-distance between an agent and an expert at the final time-step that an expert has a valid state).

to the values in other works; the low value of the displacement error may simply indicate that a majority of the scenes have unique optima. Fig. 4.4 examines the average difference in position between the agent and expert trajectories averaged across 5 training runs and demonstrates that the influence of more training data on displacement error is flat after 1000 files.

4.5.3 Policy Failure Modes

Here we investigate the mechanisms under which our policies fail to achieve their goals and collide to shine a light on potential avenues for improvement. The key failure modes we qualitatively observe are failures in scenes in which agents are required to interact with another agent either by waiting or merging. Videos of some of the failure modes can be seen at https://github.com/facebookresearch/nocturne. While we cannot measure interactivity directly, we measure a proxy by looking at the intersection of the expert trajectories. We play the experts forwards, record their trajectory as polylines, and consider a vehicle to have as many interactions as there are intersections of the vehicle's expert trajectory polyline with other vehicle's expert trajectory polylines (i.e. if two vehicles' trajectories in time cross at a point, that's an interaction). This captures interactions such as crossing at a four-way stop, merges, and others but neglects interactions such as complementary left turns and also may unintentionally pick up behaviors such as driving behind another vehicle. Close to 25% of vehicles have at least one interaction. The collision and goal rates as a function of interactions are plotted in Fig. 4.5 and demonstrate that the goal rate declines precipitously and the collision rate increases sharply as the number of interactions increases. This suggests that our agents have learned to get to their goals but perform poorly in settings where getting to the goal requires coordination with another agent.

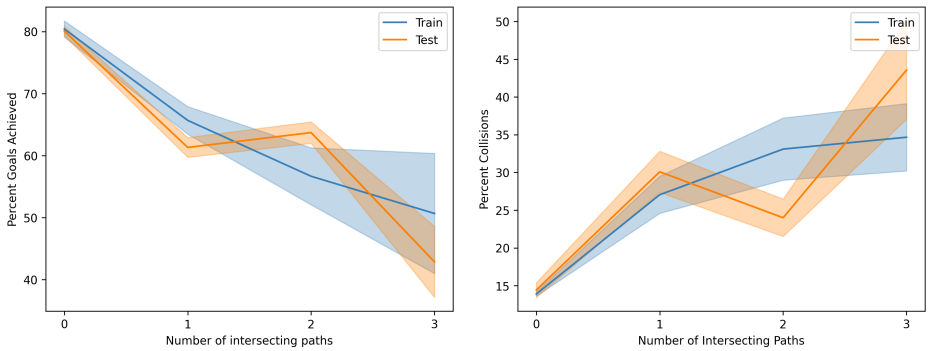


Figure 4.5: Goal rate (left) and collision rate (right) of vehicles as a function of the number of times that their corresponding expert trajectory intersected with another expert trajectory (intersections). As more than 3 interactions are rare, creating noisy statistics, scenes with more than 3 interactions are placed into the 3 interaction bin.

4.6 Conclusion

We have introduced Nocturne, a simulator and benchmark intended to aid in the study of human-like decentralized coordination for driving systems. We present results on the applications of RL and imitation to this system, however, there still remains work to be done to build agents that operate with the collision and goal rate that humans achieve as well as how to learn these agents efficiently. Given better agents, human-like rules and conventions may be emergent properties of driving safely in these settings [91].

There is also ample remaining work to be done on new benchmarks. Due to the ego-centric data collection, we are forced to remove vehicles once they achieve their goal (i.e. the last observed position of the driver in the data). However, it may be possible to use generative models or other generative mechanisms to sample new goals for the agents to continue their trajectory once they achieve the goals set out in the data. Similarly, generative models could be used to complete the traffic light states and enable the inclusion of the traffic light scenes.

Finally, one open question is how to use Nocturne agents as predictive models of human driving. At the moment a Nocturne agent requires a goal to which it is driving. To use Nocturne agents for prediction (say for an autonomous vehicle trying to predict the motion of agents in the scene), a method for inferring goals from the 1-second context needs to be implemented. A topic for future work is to use supervised methods to predict the goals and enable fully decentralized prediction of Nocturne agents from egocentric observations.

While Nocturne demonstrates the feasibility of fast, large-scale multi-agent learning on realistic driving scenarios, the framework focuses primarily on general driving coordination rather than specific traffic optimization objectives. The simulation infrastructure developed here enables the next phase of this research: applying these multi-agent learning capabilities to the specific problem of traffic wave smoothing using trajectory data collected from actual highway segments. This transition from general driving simulation to targeted traffic control

applications forms the basis for the work presented in the following chapter.

The Nocturne benchmark and all implementation files are publicly available. Due to Waymo Motion dataset licensing restrictions, trained models cannot be released, though only anonymized trajectory data is used in this work. The benchmark and all files needed to run it are publicly available at https://github.com/facebookresearch/nocturne.

Chapter 5

Deep RL Traffic Smoothing Controllers

The simulation infrastructure developments in the previous chapters have established the computational foundation for large-scale multi-agent learning, but a key challenge remains: bridging the gap between idealized simulation environments and the complex dynamics of highway traffic waves. This chapter shifts focus from throughput optimization in bottlenecks to energy efficiency in highway traffic, addressing the fundamental problem of stop-and-go wave formation that wastes substantial fuel through repeated acceleration and deceleration cycles. Rather than training controllers in simplified ring roads or hand-designed scenarios, we develop a trajectory-based learning approach that trains autonomous vehicle controllers directly on recorded highway data from Interstate 24 in Tennessee. This methodology bypasses the challenging calibration of traffic microsimulators while ensuring that learned behaviors are grounded in realistic wave dynamics, representing a transition toward data-driven controller development that prepares for deployment in Part II of this thesis.

This chapter is based on [92]: Nathan Lichtlé*, Kathy Jang*, Adit Shah*, Eugene Vinitsky*, Jonathan W. Lee, and Alexandre M. Bayen. "Traffic Smoothing Controllers for Autonomous Vehicles Using Deep Reinforcement Learning and Real-World Trajectory Data." *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2023.

5.1 Introduction

The multi-agent learning frameworks developed in the previous chapters have demonstrated the potential for autonomous vehicles to improve traffic performance through coordinated control. However, these approaches have primarily focused on throughput optimization in simplified scenarios or general coordination in complex but generic driving environments. This chapter addresses a different but equally important challenge: energy efficiency in highway traffic through the reduction of stop-and-go waves.

As discussed in Chapter 2, string instability in human driving behavior leads to the formation of traffic waves that propagate upstream, forcing vehicles through repeated acceleration

and deceleration cycles. These oscillations represent one of the most wasteful phenomena in highway transportation, converting kinetic energy into heat during braking that must be restored through fuel consumption during subsequent acceleration. This chapter investigates how autonomous vehicles can be trained to absorb these perturbations using controllers learned directly from highway trajectory data.

Transportation accounts for a large share of energy usage worldwide, with the U.S. alone attributing 28% of its total energy consumption in 2021 to moving people and goods [93]. It is the single largest sector of energy consumption, ranking over other major contributors such as the industrial sector [94]. Advances in technology have paved the way for improvements in CO_2 emissions and fuel economy via consumer shifts toward hybrid and electric vehicles (EVs), as well as innovation in vehicle technology such as turbocharged engines or cylinder deactivation [95]. Meanwhile, as autonomous vehicles (AVs) become increasingly more available in roadways, with the Insurance Institute for Highway Safety predicting there to be 3.5 million vehicles with autonomous capabilities on U.S. roads by 2025 [96], so too do their potential to yield a pronounced effect on the state of traffic, addressing the problem of energy usage from another angle.

In this era of mixed autonomy traffic, in which a percentage of vehicles are AVs with special control capabilities, a rich amount of work has been produced that shows that even a small percentage of intelligent agents in traffic are capable of achieving energy savings and traffic reduction via wave dampening or jam-absorption driving of stop-and-go waves and bottlenecks [97, 98, 99, 100]. Many studies have shown how longitudinal control of an AV can significantly impact global and local metrics such as total velocity or fuel economy; for instance, controlled vehicles are capable of completely dissipating traffic waves in a ring setting [15, 101]. To our knowledge, there have no been no large-scale tests conducted with the goal of smoothing traffic flow.

However, finding an effective way to design controllers for these settings remains an open question due to the partially observed, hybrid, multi-agent nature of traffic flow. In recent years, reinforcement learning (RL) has emerged as a powerful approach for traffic control, leveraging its ability to capture patterns from unstructured data. RL is responsible for producing quality controllers across a wide range of domains, from robotics [102] to mastering gameplay over human experts such as with Starcraft or Go [103, 104]. Within the domain of traffic, RL has been used to derive a variety of state-of-the-art controllers for improving traffic flow metrics such as throughput and energy efficiency [14, 105, 99].

With a particular focus on the real-world impacts of RL on traffic, we discuss the RL-based controller we developed for a stretch of the I-24 highway near Nashville, Tennessee. Our controllers take into account the requirements of real-world deployment, utilizing observations that are accessible via radar and cameras. Despite limited access to the full state space, our RL-based approach achieves notable fuel savings even with low penetration rates.

The contributions of this chapter are:

- The introduction of a single-agent RL-based controller developed using real traffic trajectories with advanced telemetry-enabled downstream information,

- Numerical results of the controller's performance demonstrating significant fuel savings of over 15% in scenarios exhibiting large-amplitude stop-and-go waves.

5.2 Problem Formulation

In this chapter, we consider the problem of decreasing energy consumption in highway traffic by attempting to smooth out stop-and-go waves. These waves are a phenomenon where high vehicle density causes vehicles to start and stop intermittently, creating a wave-like pattern that can propagate upstream and be highly energy-inefficient due to frequent accelerating and braking [106]. We insert a small percentage of autonomous vehicles (AVs) equipped with reinforcement learning-based controllers into the flux of traffic. Leveraging a data-driven, one-lane simulator previously introduced in [107], we simulate real-world highway trajectories. This approach is considerably more time-efficient than comprehensive traffic micro-simulations and is able to partially model the intricate stop-and-go behaviors that occur in traffic, although it overlooks complex dynamics such as lane-changing, which we rectify by incorporating a lane-changing model that we calibrate on data. The following subsections introduce the simulation as well as the different modules that it integrates.

5.2.1 Dataset and Simulation

We use the I-24 Trajectory Dataset [108] introduced in [107] along with a one-lane simulator that replays collected trajectory data. A vehicle, which we call the *trajectory leader*, is placed at the front of the platoon and replays the real-world I-24 trajectory. Behind it, we place an arbitrary number of AVs and human vehicles (introduced in Sec. 5.2.4). Typically during training, the platoon behind the trajectory leader consists of one RL-controlled AV, followed by 24 human vehicles. The goal of the AV is to absorb the perturbations in the leader trajectory, so that the energy consumption of the following human vehicles improves compared to the case where the AV is not present.

Instead of training on the whole dataset, we only train on a selected set of four trajectories containing different patterns of waves alternating between low and high speeds. This allows us to optimize directly on the dynamics we are interested in improving, without having training diluted by free-flow trajectories containing no waves for the controller to smooth. We observed that this made training faster while still yielding a controller able to generalize to unseen trajectories. As the trajectories are quite long (between 5000 and 12000 time steps, where a time step is 0.1s), each simulation randomly samples a chunk of size 500 (or 50s) within a trajectory. At evaluation time, we consider a set of six trajectories distinct from the training trajectories and simulate the whole trajectories.

5.2.2 Speed planner

We use real-time data about the state of traffic all along the I-24 in order to equip the RL control with some knowledge about the downstream state of traffic. The data is provided to us by INRIX, which broadcasts it in real time with an approximately 1-minute update interval and a 3-minute delay. The highway is divided into segments of 0.5 miles on average (with significant variance), and the data consists of the average speed of traffic in each segment. For training, we retrieved the INRIX data matching the time when the trajectories were collected on the highway (which includes delay). In the absence of such historical data, one could also generate synthetic traffic data for each dataset trajectory by artificially and averaging the trajectory speeds accordingly.

On top of this raw INRIX data, we use a *speed planner* developed in [109] that provides a profile that the controller should approximately track. The speed planner takes in the data and interpolates on the individual data points to create a continuous and less noisy speed profile of the whole highway. It then uses kernel smoothing to create a *target speed profile*, which is intended as an estimate of the speed to drive at in order to smooth out the traffic waves. However, due to delay and noisy estimates, driving exactly at this speed is insufficient to guarantee smoothing or reasonable gaps. This target speed profile, sampled at different points, is finally fed as an input to our controller, which can be used as an indication of where the free-flow and the congested regions might be.

5.2.3 Energy function

Since we aim to optimize energy consumption, we need a model that can be used to compute consumed energy in our cost functions. We use a model of instantaneous fuel consumption, a polynomial fitted on a Toyota RAV4 model that is similar to the model in [110], but with updated coefficients. The function depends on the speed of the AV and its instantaneous acceleration, as well as road grade, which we assume to be zero in this work.

5.2.4 Human Driver Behavior

To model human drivers in simulation, we use the Intelligent Driver Model (IDM) [7], a time-continuous car-following model which is widely used in traffic applications. We pick the IDM parameters such that the model is unstable below $18\frac{\text{m}}{\text{s}}$, meaning that stop-and-go waves will propagate backward in the flux of traffic and grow instead of just dissipating. Numerous results demonstrate the string-unstable qualities of human-driver behavior, both via real human drivers in real life and via models such as IDM in simulation [101].

5.2.5 Lane-changing model

We use a lane-changing model to enable more complex multi-lane dynamics for evaluation only. The model consists of a cut-in probability function $P_{\text{in}}(h, v_{\text{lead}})$ and a cut-out probability

function $P_{\text{out}}(v_{\text{lead}})$, where h is the space gap and v_{lead} is the speed of the leading vehicle. Both are piecewise second-order polynomials whose coefficients were calibrated using data collected on the I-24 highway, which has not been published yet. At each time step t, and for each ego vehicle in the simulation, P_{in} gives the probability that a vehicle cuts in front of the ego vehicle, while P_{out} gives the probability that the leading vehicle cuts out. If a cut-in happens, a vehicle is inserted such that the ratio between the space gap of the inserted vehicle and the space gap of the ego vehicle after the cut-in follows a normal distribution (also fit to data), clipped to ensure safety after insertion. This model lets us measure the robustness of the control, allowing human vehicles to cut in front of the AV as it tries to open larger gaps to smooth out traffic waves.

5.3 Controller Design

In this section, we formally define the problem in the context of reinforcement learning and discuss the structure and design of the controller.

5.3.1 Defining the POMDP

We use the standard RL formalism of maximizing the discounted sum of rewards for a finite-horizon partially-observed Markov decision process (POMDP). We can formally define this POMDP as the tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma, \Omega, \mathcal{O})$ where $\mathcal{S}$ is a set of states, $\mathcal{A}$ represents the set of actions, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ represents the conditional probability distribution of transitioning to state s' given state s and action a, $R : S \times A \to \mathbb{R}$ is the reward function, and $\gamma \in (0, 1]$ is the discount factor used when calculating the sum of rewards. The final two terms are included since the state is hidden: Ω is the set of observations of the hidden state, and $\mathcal{O} : S \times \Omega \to \mathbb{R}$ represents the conditional observation probability distribution.

5.3.1.1 Observation space

The primary observation space (at time t) consists of the ego vehicle speed v_t^{av}, the leader vehicle speed v_t^{lead}, and the space gap (bumper-to-bumper distance) h_t between the two vehicles. (Note that all distances are in m, velocities in $\frac{\text{m}}{\text{s}}$, and accelerations in $\frac{\text{m}}{\text{s}^2}$.) Two gap thresholds are also included: $h_t^{\min}$, which is the failsafe threshold below which the vehicle will always brake, and $h_t^{\max}$, the gap-closing threshold above which the vehicle will always accelerate. We also include the history of the ego vehicle's speed over the last 0.5 seconds. Finally, the observation space includes traffic information from the speed planner. This consists of the current target speed v_t^{sp}, as well as the target speeds 200m, 500m, and 1km downstream of the vehicle's current position. Note that the AV only observes its leading vehicle and that there is no explicit communication between AVs. All observations provided to the RL agent are rescaled to the range $[-1, 1]$.

5.3.1.2 Action space

The action space consists of an instantaneous acceleration $a_t \in [-3, 1.5]$. After the RL output, gap-closing and failsafe wrappers are then enforced. We define the gap-closing wrapper such that $h_t^{\max} = \max(120, 6v_t^{\text{av}})$, meaning that the AV will be forced to accelerate if its space gap becomes larger than 120m or its time gap larger than 6s. This result is then wrapped within a failsafe, which enforces safe following distances and prevents collisions. We define the time to collision Δ_t^{TTC} of the ego and lead vehicles as:

$$\Delta_t^{\text{TTC}} = \begin{cases} \dfrac{h_t}{v_t^{\text{diff}}} & \text{if } v_t^{\text{diff}} > 0 \\ +\infty & \text{otherwise} \end{cases}$$

$$v_t^{\text{diff}} = \left[v_t^{\text{av}} \left(1 + \frac{4}{30} \right) + 1 \right] - v_t^{\text{lead}}$$

where the AV velocity is slightly exaggerated to ensure robustness at both low and high speeds. The actual numbers are chosen heuristically: for instance, if both AV and leader drive at $30\frac{\text{m}}{\text{s}}$, the failsafe will ensure a minimum gap of 30m. The failsafe triggers if the time to collision ever goes below 6 seconds, in which case the RL output is overridden and the vehicle will brake at its maximum allowed deceleration. We thus have $h_t^{\min} = 6v_t^{\text{diff}}$. The final RL acceleration is given by:

$$a_t^{\text{out}} = \begin{cases} -3 & \text{if } \Delta_t^{\text{TTC}} \leq 6 \quad (\Leftrightarrow h_t \leq h_t^{\min}) \\ 1.5 & \text{if } \Delta_t^{\text{TTC}} > 6 \text{ and } h_t \geq h_t^{\max} \\ a_t & \text{otherwise} \end{cases}$$

which is further clipped to ensure that the speed v_t^{av} remains within the boundaries $[0, 35]\frac{\text{m}}{\text{s}}$. Note that the free-flow behavior due to the gap-closing wrapper will be to drive at the speed limit in the absence of a leader.

5.3.1.3 Optimization criterion

At the core, we aim to minimize the overall energy consumption of the traffic. However, as sparse rewards are harder to optimize, we employ proxies that can be minimized at each time step. We mainly aim to minimize the instantaneous energy consumption of the AV and a platoon of vehicles behind it. For comfort, and as another proxy for energy savings, we also minimize squared acceleration amplitudes. Since optimizing for energy and acceleration can be done by stopping or maintaining unreasonably large or small gaps for comfort, we penalize gaps outside of a certain range; this also penalizes the use of failsafe and gap-closing interventions. To further discourage large gaps within this allowed range, the final term adds a penalty proportional to the time gap (space gap divided by ego speed). This is formalized

as the reward function r_t, which is given by:

$$r_t = -c_1 \frac{1}{n} \sum_{i=1}^{n} E_t^i - c_2 (a_t^{\text{out}})^2 - c_3 \mathbb{1}\left[h_t \notin [h_t^{\min}, h_t^{\max}]\right] - c_4 \frac{h_t}{v_t^{\text{av}}} \mathbb{1}\left[h_t > 10 \wedge v_t^{\text{av}} > 1\right]$$

where E_t^i is the instantaneous energy consumption of vehicle i at time t, where index $i = 1$ corresponds to the AV, and indexes $i = 2$ to $i = n$ correspond to the following $n - 1$ IDM vehicles. $\mathbb{1}$ is defined such that $\mathbb{1}[\mathcal{P}] = 1$ if $\mathcal{P}$ is true, 0 otherwise.

5.3.2 Training algorithm

We use single-agent Proximal Policy Optimization [25] (PPO) with an augmented value function as our training algorithm. PPO is a policy gradient algorithm, a class of RL techniques that optimize for cumulative, discounted reward via shifting the parameters of the neural net directly. More explicitly, policy gradient methods represent the policy as $\pi_\theta(a|s)$, where θ are the parameters of the neural net.

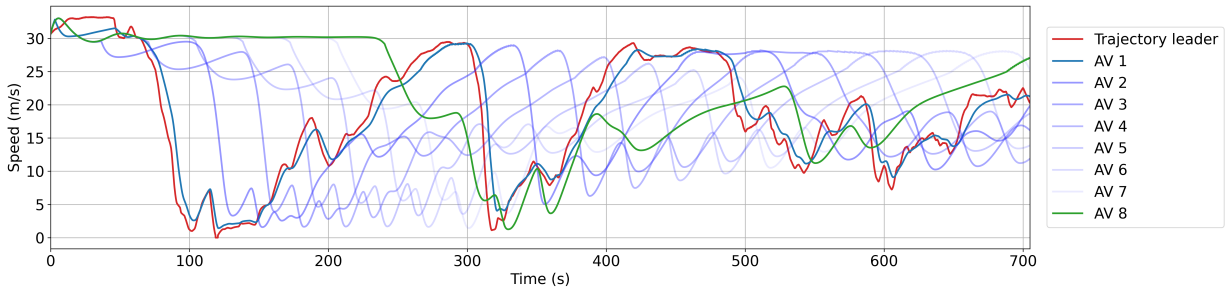


Figure 5.1: Evolution of the speed of the trajectory leader and the AVs in a platoon of 200 vehicles. The 8 AVs are equally spaced at a 4% penetration rate. The first AV in the platoon is shown in blue, the following ones are displayed by decreasing opacity and the last one is in green, demonstrating the smoothing effect of the AVs on the leader trajectory. In particular, one can see how the first AV (in blue) already smoothes the trajectory leader (in red), doesn't slow down as much or accelerate as fast, and thus saves energy.

During training, we give additional observations that are available in simulation as an input to the value network. This includes the cumulative miles traveled and gallons of gas consumed, which allows for estimating energy efficiency. This augmented observation space also includes the following metadata: the size of the finite horizon, an identifier for the specific trajectory chunk being trained on, and the vehicle's progress (in space and time) within this chunk. The additional information removes some of the partial observability from the system, allowing the value function to more accurately predict the factors that influence reward.

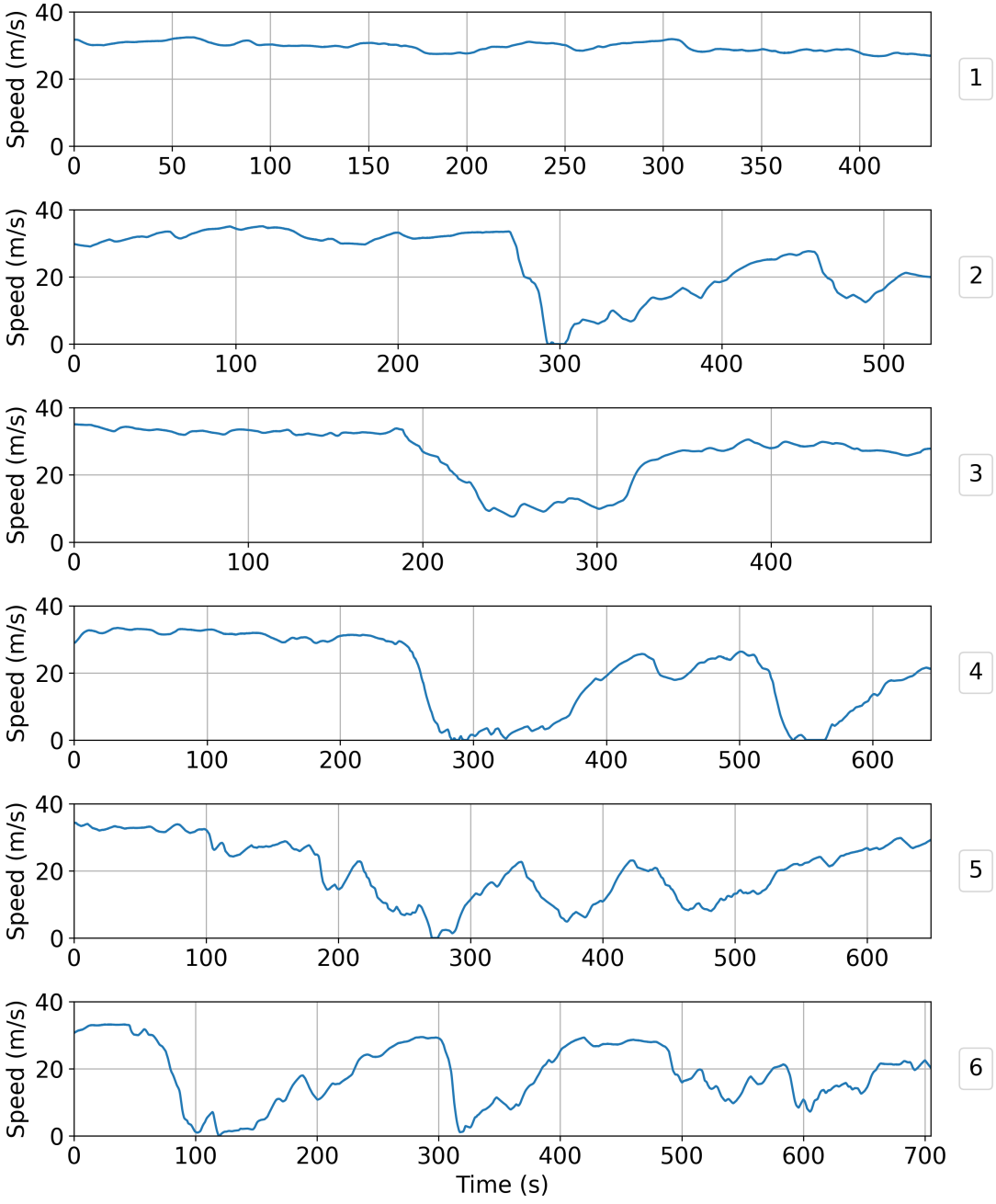


Figure 5.2: Trajectories used for evaluation, numbered from 1 to 6. The first one corresponds to free flow, while the five others contain both low and high speeds, including sharp breaking, sharp accelerating or stop-and-go behaviors.

5.3.3 Experiment details

We run the PPO experiments using the implementation provided in Stable Baselines 3[1] version 1.6.2. We train the model for 2500 iterations, which takes about 6 hours on 90 CPUs. We use a training batch size of 9000, a batch size of 3000, a learning rate of $3 \cdot 10^{-4}$ and do 5 epochs per iteration. The simulation horizon is set to 500, and we do 10 simulation steps per environment step, ie. each action is repeated 10 times. Given that the simulation time step is $dt = 0.1$s, this means that the action changes only every second during training. This allows us to artificially reduce the horizon so that it only is 50, meaning that each training batch contains 100 simulation rollouts. The agent's policy is modeled as a fully-connected neural network with 4 hidden layers of 64 neurons each and tanh linearities, with continuous actions. The value network has the same architecture as the policy network. We set the discount factor γ to 0.999, the GAE value λ to 0.99, and the other training and PPO parameters are left to their default values.

We train with a platoon of $n = 25$ vehicles (not including the leading trajectory vehicle) consisting of one AV followed by 24 IDM vehicles. For our reward function, we used coefficients $c_1 = 0.06$, $c_2 = 0.02$, $c_3 = 0.6$, $c_4 = 0.005$. Both the model parameters and training hyperparameters are determined through grid search, with each experiment conducted using 4 to 8 distinct random seeds to overcome instances of the agent getting trapped in local optima.

5.4 Results

In this section, we analyze the performances of our RL controller in simulation, in terms of energy savings, wave-smoothing, behavior and robustness.

The smoothing effect of the AVs is illustrated in Fig. 5.1, where one can see the speeds of all the AVs in a platoon of 200 vehicles as a function of time on trajectory 6 (see Fig. 5.2). One can observe how the speed profiles become smoother and smoother after each AV.

Fig. 5.3 illustrates the smoothing performed by the RL agents on trajectory 6 in a different way. The top-left time-space diagram shows that the humans don't smooth any waves; on the contrary, they even create some due to the string-unstable nature of the IDM we use. At a 10% penetration rate, most of the waves get smoothed out, less at 4%, and even less with lane-changing. However, in all 3 cases, the diagrams clearly demonstrate the improvement over the baseline. As expected, AVs opening larger gaps also leads to decreased throughput, and the best throughput is achieved when the lane-changing model is enabled and human vehicles fill in the gaps. This comes down to a trade-off between throughput reduction and energy savings, which can be tuned by varying the penetration rate.

Table 5.1 shows the energy savings that our controller achieves on the evaluation trajectories (shown in Fig. 5.2) when deployed on AVs at two different penetration rates, with and without lane-changing enabled. The percentages correspond to how much the average system

[1] https://github.com/DLR-RM/stable-baselines3

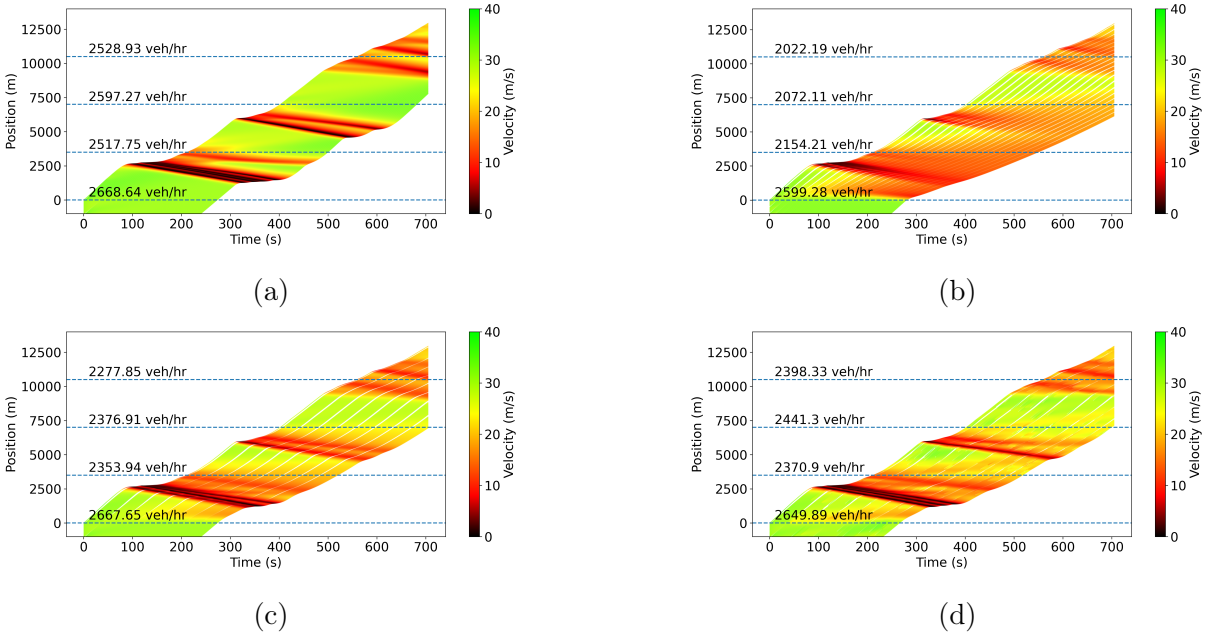


Figure 5.3: Time-space diagrams, each representing a simulation of 200 vehicles. Each vehicle trajectory is plotted as a line in position-by-time space, with color representing the speed of that vehicle. One can observe the wave-smoothing effect of the RL-controlled AVs over time. Horizontal lines display the throughput of the traffic flow at that particular position. Also note that as a warm-up, all AVs behave as humans when their position is negative. **Top left:** all 200 vehicles are IDMs. Traffic waves are illustrated by the red and black colors, while bright green represents free flow. **Top right:** 20 equally-spaced RL-controlled AVs (10% penetration rate, 1 AV every 10 vehicles). One can see the larger gaps opened by the AVs as the white lines between platoons. **Bottom left:** 8 equally-spaced RL-controlled AVs (4% penetration rate, 1 AV every 25 vehicles). **Bottom right:** 8 equally-spaced RL-controlled AVs (4% penetration rate) with the lane-changing model enabled (note that it is disabled in all 3 other subfigures).

Index	10% w/o LC	10% w/ LC	4% w/o LC	4% w/ LC
1	+7.33%	+8.39%	+4.29%	+6.12%
2	+10.87%	+12.63%	+7.04%	+9.42%
3	+14.65%	+14.48%	+9.02%	+7.23%
4	+15.02%	+13.19%	+9.23%	+8.54%
5	+22.58%	+15.77%	+17.05%	+8.37%
6	+28.98%	+18.55%	+19.96%	+15.40%

Table 5.1: We run simulations with 200 vehicles, and show the improvement in system MPG when we control 10% (Left) or 4% (Right) of the vehicles, compared to when all vehicles behave as IDMs, with and without lane-changing in both cases. The trajectories used for evaluation can be seen in Fig. 5.2, with corresponding indexes.

miles-per-gallon (MPG) value increases when AVs use our RL controller, compared to when they all behave as humans. The average MPG is defined as the sum of the distances traveled by all the vehicles in the simulation, divided by the sum of their energy consumptions.

We can observe the energy savings varying a lot depending on the trajectories, which is expected since trajectories that are mostly free flow (like trajectory 1) cannot be improved much, while one with a lot of stop-and-go waves (like trajectory 6) has a lot of potential to be smoothed. As one can expect, energy savings decrease as the AV penetration rate decreases or as lane-changing is enabled, but even at 4% penetration and with lane-changing, the controller reduces the energy consumption on trajectory 6 by over 15%, while only reducing throughput by 5%.

We also note that the controller appears robust to not having access to the speed planner. For example at a 4% penetration rate and without lane-changing, the control achieves +16.87% energy improvement on trajectory 6 without the speed planner (compared to +19.96% with the speed planner), and the trend is similar on the other trajectories.

Finally, in Fig. 5.4, we illustrate the gaps opened by the AV on trajectory 6, along with the failsafe and gap-closing thresholds. The gap-closing threshold allows the AV to open larger gaps and consequently absorb abrupt braking from its leader while ensuring that these gaps are not overly large. As can be expected, we have observed that the larger the maximum gap we allow, the better the AV performs in terms of energy savings. However, a larger maximum gap is usually accompanied by a decrease in throughput, which is again a trade-off. The failsafe threshold mostly ensures safety and comfort for the driver, although it is worth noting that when deploying the controller, we integrate an additional explicit safety wrapper, as detailed in [107].

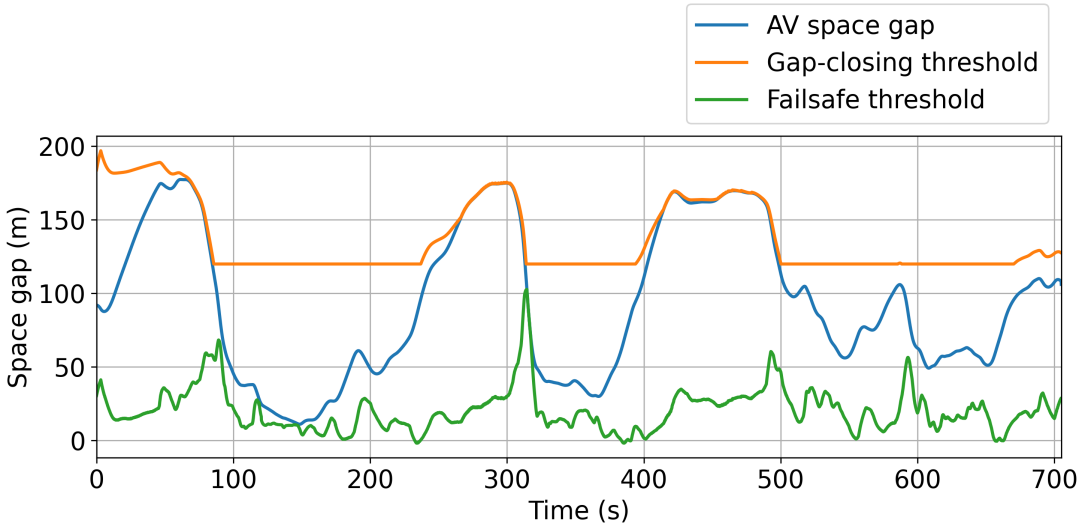


Figure 5.4: Space gap of the first AV in the platoon (in blue) by time, on the same scenario as in Fig. 5.1. The orange line shows the gap-closing threshold $h_t^{\max}$ and the green line shows the failsafe threshold $h_t^{\min}$, introduced in Sec. 5.3.1.2.

5.5 Conclusion

In this work, we developed RL policies that incorporate both accessible local observations and downstream traffic information, achieving substantial energy savings in simulation. There are several avenues for future research. Given that all the training is conducted in a simulated environment, it would be beneficial to train the agent with more realistic dynamics by enhancing the accuracy of the various models that make up this simulation, like human driving behavior, lane-changing dynamics, and energy consumption metrics. Additionally, it would be interesting to explore multi-agent RL to help the model be robust to interactions between AVs and potentially enable cooperation between them.

While our simulation process has a distinct speed advantage over large micro-simulators, it could benefit significantly from vectorization. Moreover, despite our technical capacity to deploy our controller safely onto a real-world vehicle, gathering results is challenging and necessitates further field tests. Another direction of research we are exploring consists of training and deploying adaptive cruise control (ACC)-based controllers, where the policy outputs a desired set-speed instead of an acceleration. By design of the ACC, the control would be safe and smooth, and easily deployable at a large scale simply by augmenting the onboard ACC algorithm to use the RL control as a set-speed actuator.

The trajectory-based training approach developed in this chapter represents a crucial step toward bridging the simulation-to-deployment gap in autonomous vehicle traffic control. By demonstrating that controllers can achieve substantial energy savings when trained directly

on highway data, this work establishes the methodological foundation for actual deployment on production vehicles. However, significant challenges remain in translating these simulation results to operational systems that must function safely and reliably in unpredictable traffic conditions. Part II of this thesis addresses these deployment challenges, beginning with the complete pipeline from training to field validation in the following chapter.

Part II

Field Validation: 100-Vehicle Highway Deployment

Chapter 6

Deploying Traffic Smoothing Cruise Controllers

Part I of this thesis established the theoretical and simulation-based foundations for autonomous vehicle traffic control, progressing from multi-agent bottleneck optimization through scalable simulation infrastructure to trajectory-based wave smoothing. Part II now addresses the critical challenge of translating these simulation successes to deployment on production vehicles in highway traffic. This chapter presents the first step in this validation process: developing and demonstrating a complete pipeline that enables controllers trained on trajectory data to operate safely on physical vehicles without retuning or recalibration. By constructing a dataset of highway trajectory data from Interstate 24 and training controllers directly on this data, we bypass the intractable problem of calibrating high-fidelity traffic microsimulators while ensuring learned behaviors are grounded in the traffic dynamics of the target deployment environment. The successful field validation presented in this chapter demonstrates that the simulation-to-deployment gap can be effectively bridged through trajectory-based learning, establishing the foundation for the large-scale field experiments that follow.

This chapter is based on [107]: Nathan Lichtlé*, Eugene Vinitsky*, Matthew Nice*, Benjamin Seibold, Dan Work and Alexandre M. Bayen. "Deploying Traffic Smoothing Cruise Controllers Learned from Trajectory Data." *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

6.1 Introduction

The trajectory-based controllers developed in Chapter 5 demonstrated substantial energy savings in simulation by training directly on highway data rather than calibrated microsimulators. However, a fundamental question remains: can these simulation results translate to performance on production vehicles operating in unpredictable highway traffic? This chapter addresses this critical simulation-to-deployment gap by presenting a complete pipeline that

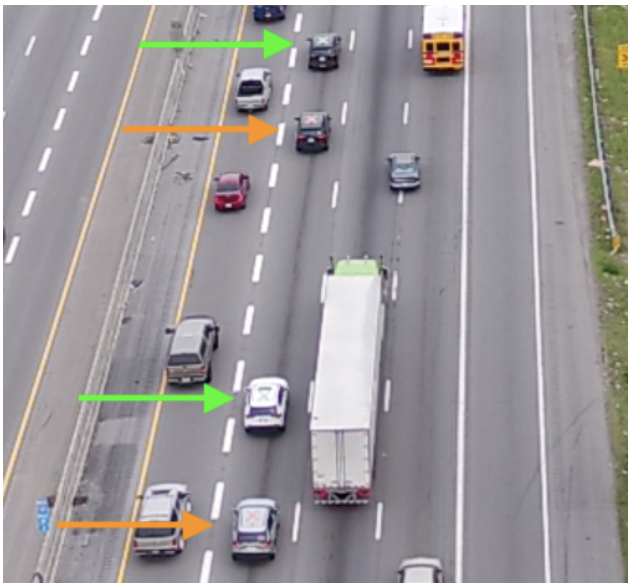


Figure 6.1: 4 of 11 vehicles in formation on the roadway. Green arrows and green X on roof indicate AV (AV), orange arrows and orange X on roof indicate human driven sensing vehicle (H). During experiments platoon formed in this order: [H, H, AV, H, AV, H, AV, H, AV, H, H], with no control over traffic flow consistently cutting in and out.

enables controllers trained on trajectory data to operate safely on physical vehicles without retuning or recalibration.

The challenge of deploying learned traffic controllers stems from the difficulty of accurately modeling the complex, stochastic nature of highway traffic. Traditional approaches that rely on hand-crafted microsimulators often fail to capture the nuanced wave dynamics and human driving behaviors that autonomous vehicles must handle in deployment. By training controllers directly on recorded trajectory data from the target highway and then deploying them zero-shot, we bypass these modeling challenges while ensuring that learned behaviors are grounded in the traffic dynamics of the actual deployment environment.

In particular, prior work [15] has shown that even at low penetration rates of less than 4%, empirical and theoretical evidence suggests that AVs can significantly reduce stop-and-go traffic, a pernicious transitory phenomenon in which vehicles alternate between starting and stopping, consuming extra fuel in the process. However, prior approaches have a unifying problem: they are developed and analyzed in simplistic settings such as rings or hand-designed input perturbations. Testing on more complex settings is difficult as: 1) real-world highway sensor data are sparse and lack required resolution and detail needed for accurate modeling; 2) developing simulators that properly reproduce emergent structures from many-vehicle-interactions is challenging.

Building more complex models is heavily data constrained. Loop detectors only yield macroscopic statistics, while cameras tend to cover only a small portion of the roadway. This

lack of available data is a fundamental issue as the trajectories of vehicles traveling through waves depends on the wave speed [111], and yet the wave speed is difficult to estimate with available stationary sensors. However, without an accurate means of reconstructing the stop-and-go traffic that is likely to occur on a particular highway, it is difficult to validate how a controller will perform when deployed on that highway. Consequently, it is unclear whether progress on control design for real-world smoothing is being made.

The contribution of this chapter is a pipeline which avoids these aforementioned modeling challenges and produces a reinforcement learning (RL) controller that is then successfully deployed on four vehicles in dense highway traffic. This pipeline has three parts: (i) the data collected from human driving trajectories, (ii) the RL controller, and (iii) the deployment of the controller on physical vehicles.

We are able to avoid modeling challenges by learning a traffic-smoothing controller directly from data collected from human driving trajectories. Instead of attempting to build a high fidelity simulation, we evaluate and train our controllers on collected highway trajectory data, ensuring that our controllers are learning to smooth a realistic representation of waves from the particular highway on which we intend to deploy AVs. We construct a simplified controller evaluation procedure in which a simulated mixed platoon of AVs and human drivers follows directly behind trajectories collected by a human driver on I-24, an interstate highway in Tennessee, scoring the controllers by their ability to improve energy consumption while maintaining traffic throughput. This approach sidesteps the aforementioned difficulties in calibrating both the waves and the microscopic car following dynamics. Using Proximal Policy Optimization[27], an RL policy gradient algorithm, we learn a controller that decreases the fuel consumption of the platoon in simulation by 16% for the AV and 10% on average for the platoon vehicles. Finally, we deploy the controller on real vehicles in highway traffic, showing the viability of this controller to create real-world energy savings and use of the complete pipeline.

The rest of this chapter is organized as follows: in Section 6.2 we discuss works informing this research, in Section 6.3 we discuss the data collection, cleaning, and analysis, in Section 6.4 we discuss the controller design and structure, algorithm, training details, and deployment pipeline, in Section 6.5 we discuss the simulation results, and experimental results, and finally in Section 6.6 we discuss and provide practical considerations to be considered in future work.

6.2 Related Work

Prior work has investigated the efficacy of traffic smoothing controllers on settings such as rings or hand-designed input perturbations. The most closely related works are [15, 112, 113]. In [15], the authors showed that a single AV could be used to dampen stop-and-go waves on a ring with 21 human drivers, yielding sharply improved fuel efficiency. The work in [112] studies traffic smoothing with connected AVs and demonstrates that the connectivity can be used for more effective dampening of waves on a single-lane, eight-mile-long public road. The work in [113] conducted an experiment in which individual vehicle speeds were controlled to

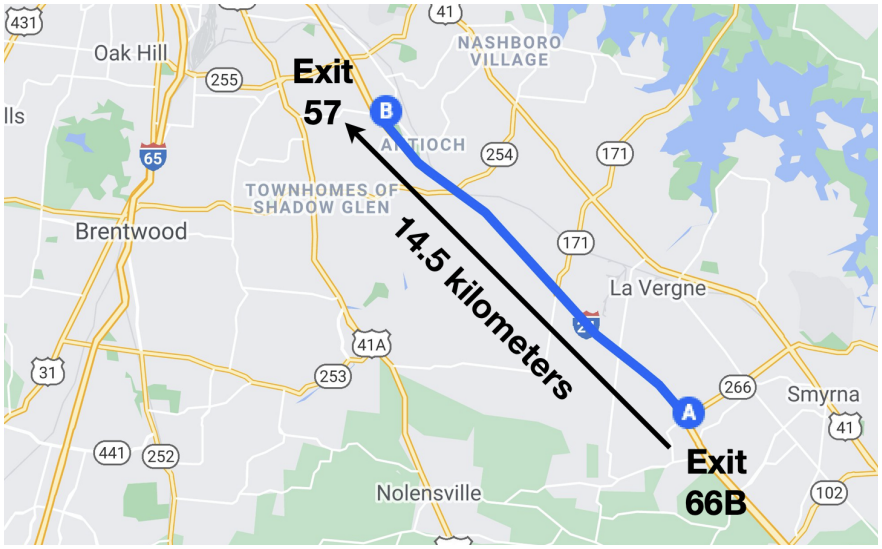


Figure 6.2: Portion of the I-24 highway on which we collected most of the dataset (section 6.3.1) and where we ran the experiments described in section 6.5.

smooth traffic flow. Other works have considered the wave dampening properties of existing commercially-available cruise controllers, with [114],[115],[116] all observing that the vehicles they tested were string unstable. Finally, [117] has studied some of the sim-to-real challenges in deploying RL-learned cruise controllers into more realistic settings.

Prior work has also studied the use of reinforcement learning (RL) and optimal control for developing micro-level controllers that optimize mixed autonomy traffic. [118] learns memory-based policies that infer ring densities and consequently outputs near-optimal policies for the ring, [119] uses multi-agent reinforcement learning (MARL) to optimize the throughput of a merging region, and [120] employs MARL to investigate the potential impacts of altruistic autonomous driving on a merge scenario. At a network level, RL has been used to learn routing behaviors for AVs that induce the human drivers to select paths that lead to decreased congestion [121].

6.3 Training Set

Here we detail the human driver data collection procedure. The data serve as the basis for which we train wave smoothing controllers. We then briefly describe the data cleaning process and analyze the distribution of trajectories collected.

6.3.1 Data Collection

We collect data by recording trajectory data on a 14.5-kilometer-long segment (displayed in Fig. 6.2) of I-24 located southeast of Nashville, Tennessee. Each drive is conducted in an instrumented vehicle that logs CAN data via libpanda [122] and GPS data from an onboard receiver. Collected measurements from the vehicle CAN data include the velocity of the *ego vehicle* (the vehicle being driven), the relative velocity of the *lead vehicle* (the vehicle in front of the ego vehicle), the instantaneous acceleration, and the *space-gap* (bumper-to-bumper distance).

The drives are varied in the time of day, day of the week, direction of travel on the highway, and level of congestion. Each drive is made up of one or more passes through the highway stretch of interest. The data used to train the algorithm in this chapter are made publicly available at [123], along with more details on the data.

6.3.2 Data Cleaning

The raw data for a given drive were recorded in two files: a CAN data file and a GPS file. The pertinent data are pulled from the CAN data and interpolated to the GPS time, which is measured at 10 Hz. High-frequency CAN data are down-sampled and linearly interpolated to match the GPS time, and low-frequency CAN data undergo linear interpolation to match the 10 Hz GPS time as well. Distance traveled and direction of travel are computed using the GPS position data. Since the westbound data contain more regular congestion, we focus on westbound data for training. The westbound data contain 60 trajectories, representing 8.8 hours and 772.3 kilometers of driving.

6.3.3 Dataset Analysis

The data are collected over a wide range of traffic conditions ranging from congested traffic that is nearly stopped to free-flow, max-speed traffic, including many acceleration and deceleration patterns corresponding to stop-and-go traffic. Fig. 6.3 shows an example velocity and space-gap profile from a trajectory in the dataset, where we can observe the ego vehicle going quite rapidly from low to high speeds. While our main interest is in smoothing high-frequency waves, which occur primarily in congestion, the distribution of speeds in the training dataset tends towards higher speeds. While we could filter the dataset to only contain low speeds, likely making the learning problem simpler, Fig. 6.3 suggests that regions of congestion are often quickly followed by regions of high speed. To ensure our controller behaves appropriately at high speeds and in transitions between high and low speed regions, we keep both low and high velocities in the training dataset.

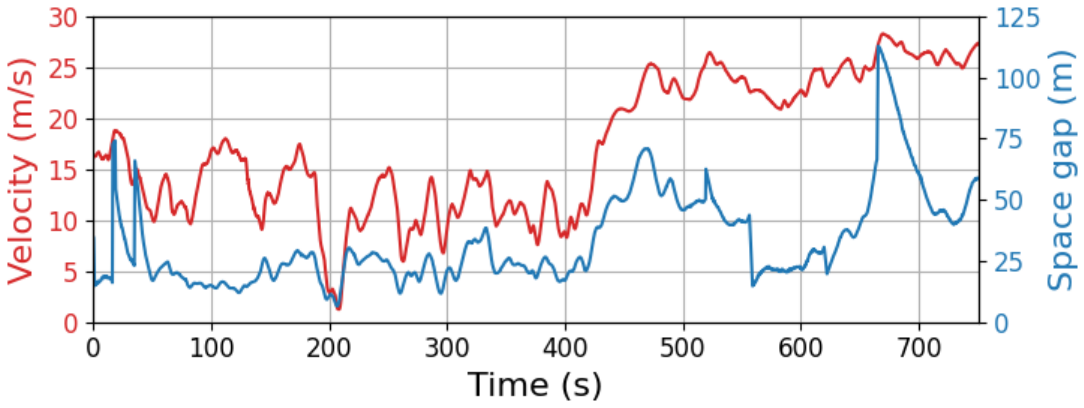


Figure 6.3: Velocity of the ego vehicle (blue) and space-gap to the lead vehicle (red) for a single trajectory in the dataset, containing sharp variations in both velocity and space-gap.

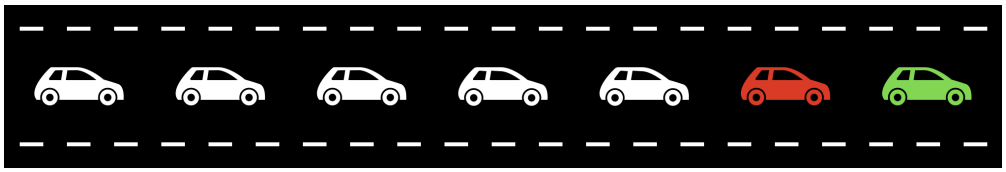


Figure 6.4: Vehicle formation used in simulation. A trajectory leader (in green) driving a speed profile drawn from the dataset is placed in front of an AV (in red) which is followed by a platoon of 5 human vehicles (in white), modeled using the Intelligent Driver Model.

6.3.4 Constructing the Training Environment

In order to use the collected data, we build a one-lane training environment where the AV follows behind the trajectory collected from the human drivers. The human driver is placed at the front of a simulated platoon, followed by the AV, followed by five vehicles driving according to the Intelligent Driver Model (IDM) [7] with a set of parameters that are string unstable below $18\frac{\text{m}}{\text{s}}$, which ensures that the waves grow in congestion. Although having a full micro-simulation of the I-24 would allow for training on a model with complex long-range interactions between the vehicles, the simulator proposed here allows us to train on realistic driving dynamics that are representative of both the types of waves on this highway and how drivers react to wave formation. As an additional benefit, this single-lane simulation using half a dozen vehicles achieved 2000 steps per-second while a comparable micro-simulation of the full 14 kilometer road section would have thousands of vehicles in congestion and would be very computationally costly to evaluate.

Our collected dataset contains both the trajectory of our drivers and the vehicles in front

of them (via space-gap and relative velocity data logged on the CAN). We discard the lead trajectories and do not use them for simulation as the lead trajectories contain both cut-ins (a vehicle cuts in between the lead vehicle and the ego driver) and cut-outs (the lead vehicle changes lanes). While cut-outs are likely unaffected by the behavior of the ego driver, cut-ins are likely a function of the spacing between ego driver and lead vehicle. Since our trained controller will have different space-gap keeping patterns, it is possible that the observed cut-outs would not occur given the controller's choice of space-gaps; to avoid dealing with this counterfactual we simply do not use the leader data for training and only keep ego vehicle data for our lead trajectories. Since the human drivers who collected the dataset intentionally rarely change lanes, our simulator consequently does not contain lead-vehicle lane changes. Finally, we note that we do not split the data into a train and test set; we train our controller on all of the available trajectories and instead use the deployment as our test set.

6.4 Method

In this section we describe the control design and structure, the details on how the controller is trained on the trajectory data, and the deployment pipeline that enables experiments to be conducted on a real vehicle platform.

6.4.1 Controller Design

For the model of the system dynamics, controls, and inputs we adopt the following system. As it is unclear whether the Markov property holds for this system [124], we will assume that the system described below may be slightly non-Markovian.

State space $[v, v_{\text{lead}}, h]$ where v is the AV speed, v_{lead} the speed of the vehicle right in front of it, and h the space-gap. All of these features can be acquired by using the forward-facing radar and the data collection software[122, 125] that we place on our vehicles.

Action space an instantaneous acceleration a, bounded between $[-4.5, 2.6]$ $\frac{\text{m}}{\text{s}^2}$, to be applied to the AV. Note that we do not allow the AVs to lane-change in this chapter.

Reward function the reward the AV receives at time-step t is a combination of minimizing energy consumption, acceleration regularization and penalties for leaving too small or too large gaps. It is given by

$$r_t = 1 - c_0 E_t - c_1 a_t^2 - c_2 P_t \ .$$

Here E_t is the instantaneous gallons of fuel consumed by the AV (given by a piece-wise polynomial energy model calibrated to a RAV-4 Toyota vehicle; the fitting procedure and function coefficients are given in [126]), a_t the AV's instantaneous acceleration in $\frac{m}{s^2}$ and P_t its gap penalty, all at time-step t. The first term is intended to discourage fuel consumption, the second to encourage smooth driving, and the third to discourage the formation of large

gaps that induce cut-ins or small gaps that might lead to driver discomfort. For our reward functions, we use coefficients $c_0 = 1.0\frac{1}{\text{Gal}}$, $c_1 = 0.002\frac{s^2}{m}$ and $c_2 = 2$, and penalize with $P_t = 1$ when the gap is below 7m, above 120m or when the time-gap (i.e., space-gap over speed) to the leader is below 1 second. These particular values were selected via an informal hyperparameter search and found to yield improved fuel consumption of the platoon while maintaining all constraints that might set off the penalty term P_t.

Finally, we note that our reward function does not include the energy consumption of the following platoon. While we experiment with such a reward, we observed more improvement by only optimizing for the energy consumption of the AV.

6.4.2 Controller Structure

The RL controller $G(\cdot)$ takes as inputs the current vehicle speed, the speed of the lead vehicle, and the space-gap provided by recorded or real-time CAN-to-ROS translation[125], and outputs a desired acceleration to a supervisory FollowerStopper[127] wrapper controller. The FollowerStopper leverages reachability analysis to verify safety and allows for total avoidance of a collision with the lead vehicle by taking in a desired velocity and returning a safe commanded velocity v_{safe}. The controller output during learning is acceleration-based; to convert it into a desired velocity we return $v_{\text{des}} = v_t + 0.6 \cdot G(\cdot)$ where v_{des} is the speed passed to the FollowerStopper and $G(\cdot)$ is the acceleration output by the RL controller. This desired velocity is then sent via CAN[122] to the vehicle's ECU for actuation. The particular "integration constant" 0.6 corresponds to the vehicle's responsiveness of about $\tau = 0.6s$, which is found by making the mapping from desired speed to realized speed as close as possible to the identity function, a mapping that we get from a transfer function approximating the vehicle's dynamics.

6.4.3 Algorithm

We train our policy using Independent Proximal Policy Optimization [27] (PPO), a policy gradient algorithm. We modify the standard PPO algorithm by providing the value function with a few additional inputs: the total distance traveled from start to time t, the total energy consumed by the agent at time t, and time t. The value function V^π estimates the discounted cumulative reward from a given state s_t and a particularly controller π. This quantity is difficult to estimate without the additional information we provide due to the partially observed state described in Sec. 6.4.1. The non-local information provided to the value function is used exclusively during training for variance reduction (see [27] for details), and these additional inputs are neither available nor needed by the controller during evaluation.

Training was done using the PPO implementation provided in Stable Baselines 3 [128] version 1.0, a Pytorch-based deep RL library. Training details and hyperparameters are provided in the linked code-base.

6.4.4 Deployment Pipeline

An initial software-in-the-loop (SWIL) step is taken to check functional correctness and interface testing in a 2-vehicle Gazebo simulation [129]. Structured velocity profiles (e.g., constant acceleration, sinusoidal, trapezoidal) are input to the controller to ensure outputs are not unusual. This also checks for software correctness. For hardware-in-the-loop (HWIL) deployment on the physical vehicle, there is a series of three tests to mitigate safety risks from the transition from simulation to physical vehicle before testing the controller on the I-24 segment. All three tests have varied input from the leader vehicle to ensure performance in non-equilibrium states.

First, the controller is tested in a 'Ghost Mode' as in [130] where the vehicle follows a simulated 'Ghost' vehicle as its leader. This provides the opportunity for a bad implementation to fail and crash into a virtual vehicle instead of a real one. The full HWIL setup is used with the modification that the real sensing done by the vehicle is replaced by a spoofed recording of a lead vehicle ahead using [125]. Second, the controller is tested in a 'CAN Coach Mode' as in [130] where the controller feedback is sent through a human-in-the-loop (HIL) for actuation. This second test occurs on a low-traffic, high speed route. Here the vehicle sensors feed real-time data into the controller, and the controller gives feedback to the HIL to indicate what input should be provided to the vehicle, but if the controller provides unsafe input to the HIL it is rejected to maintain safety and replaced with human control.

Finally, the controller is used on a low-traffic, high-speed route testing the complete HWIL control loop. Once these are successfully finished, the controller is ready to be tested on the heavy traffic, high speed I-24 roadway segment.

6.5 Results

6.5.1 Simulation Results

Here we analyze the performance of the controller in terms of energy efficiency improvements in miles per gallon (MPG) observed in our simulator. In Fig. 6.6, we compare the energy consumption of the AV and all vehicles in the platoon (as shown in Fig. 6.4) when the AV is using our RL controller compared to an IDM controller, over the whole training dataset. We split the trajectories by leader speed, computing the energy savings at leader speeds above and below $18\frac{\text{m}}{\text{s}}$, which is the speed boundary beyond which IDM vehicles with the parameters used in this chapter go from being string-unstable to string-stable. The results in the left and middle columns indicate that most of the expected energy improvements from the controller will come at low speeds. While these savings are significant, in more complex settings imperfections in actuation, modeling of human drivers, and cut-ins would likely lower the actual improvement. The rightmost column is described in Sec. 6.5.2.

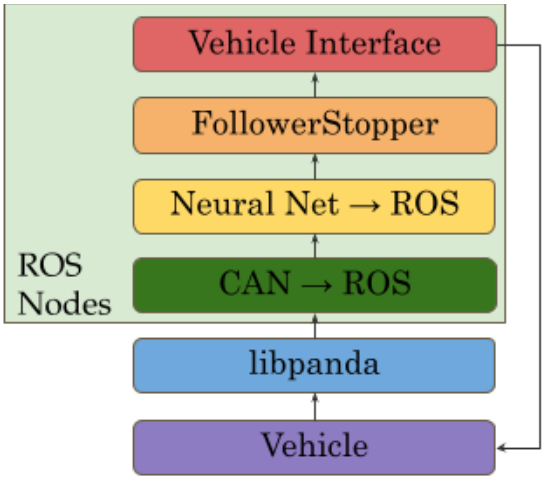


Figure 6.5: Diagram showing how information flows though the HWIL system when deployed. The vehicle sensors send data on the CAN bus. Libpanda[122] records the data and data are translated into ROS [125]. The neural net is embedded in a ROS node subscribing to pertinent data, and its output is filtered through a supervisory safety controller to get v_{safe}. This value is sent to the vehicle interface which takes a desired ROS command and sends it via CAN to the vehicle.

6.5.2 Experimental Results

In this section, we describe the validation experiment conducted on the segment of I-24 shown in Fig. 6.2. We assess the success of the controller deployment onto AVs by showing an accurate match between simulation and reality. Finally, we seek to determine whether our controller improved the energy efficiency of its platoon.

Fig. 6.1 shows four vehicles from the eleven-vehicle platoon of alternating humans and AVs that we deployed on I-24.

For each test, we got the platoon onto the highway without any non-platoon vehicles lane-changing into it. Once on the highway, non-platoon traffic cut in and out of our platoon. Since our vehicles were only instrumented to sense the vehicle in front of them, the number of vehicles that managed to enter into our platoon is unknown. We ran experiments on August 2nd, 4th, and 6th of 2021, each day launching the platoon of vehicles three times and bringing the vehicles back to the start of the highway section in between each run. The controller presented here was only actuated on 08/06, over three tests that occurred at 6:45, 7:29 and 8:36 AM. Fig. 6.7 shows individual vehicle trajectories on a time-space diagram from the 6:45 AM test; the two regions of red correspond to congestion events. The deployment of the controller from simulation to real vehicles was overall successful as all tests ran safely and smoothly.

We investigate the effect of the sim-to-real gap induced by the presence of cut-ins and

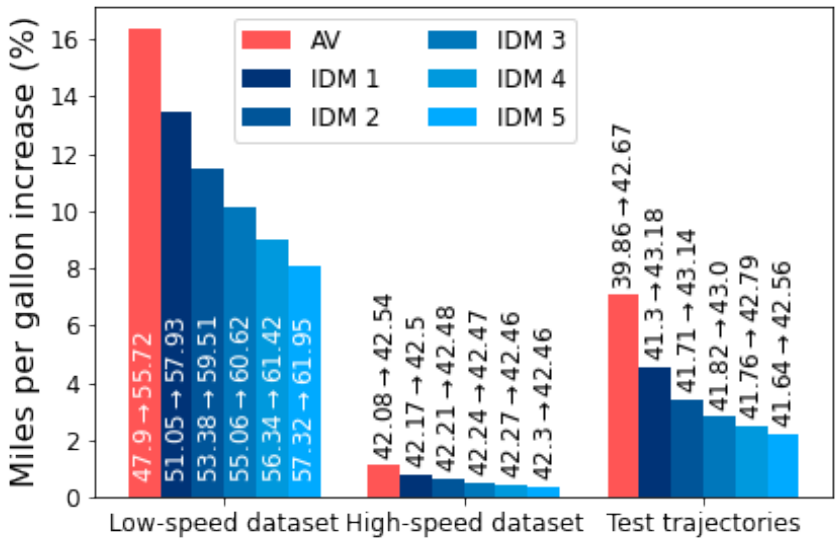


Figure 6.6: Percent improvement in MPG relative to a baseline in an IDM vehicle leads the platoon in Fig. 6.4. Each column contains both percent improvement on the y-axis and MPG values used to compute this improvement inside each column with IDM (AV) on the left (right) of the arrow. High and low speed columns are over the training set. The "Test trajectories" column is the controller evaluated on data from the physical test.

cut-outs, which we did not have when training our controller, as well as imperfect modeling of the transfer function of the AV. First, we attempt to compute a counterfactual baseline in which we replay our controller in simulation behind a trajectory collected during the tests. This mechanism is imperfect as the real-world trajectory has cut-ins and replaying a different controller behind it might affect the cut-in frequency. Without a model of lane changing, we cannot perform this counterfactual perfectly so instead we make the calculations assuming that both the times when cut-ins occur and the space-gap directly after the cut-in are unchanged. Occasionally, we choose to relax this latter condition in order not to experience, in simulation, cut-ins that would be more aggressive than what the real-world AV experienced. To that end, at each time-step t where a cut-in would leave the AV with a space-gap h_t^{sim} while the real-world AV experienced a space-gap h_t^{real}, we set $h_t^{\text{sim}} = \max(h_t^{\text{sim}}, \min(h_{t-1}^{\text{sim}}, h_t^{\text{real}}))$.

Fig. 6.8 shows the velocity and time-gap (space-gap divided by velocity) of an AV from the validation experiment as well as the replay of the trajectory in our simulation using the counterfactual cut-in mechanism mentioned above. The velocity profile of the vehicle closely matches its expected behavior computed in simulation. Although there are mismatches around cut-ins and cut-outs (regions where time-gap changes discontinuously), the time-gaps are relatively close and we can observe the vehicle roughly tracking a three-second time-gap in both cases. We observe similar results on the other trajectories we collected during the tests.

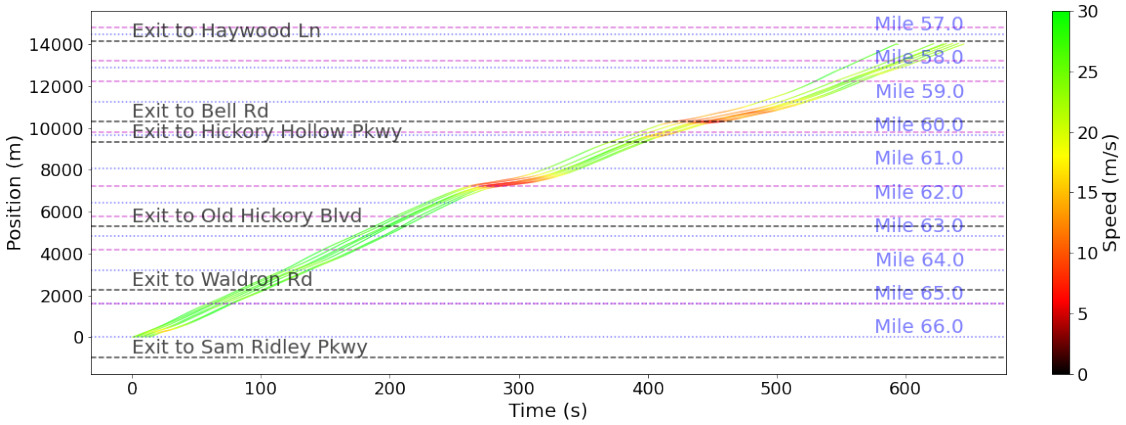


Figure 6.7: Time-space diagram showing the trajectories of our platoon of vehicle during the first test. We can observe two low-speed regions of congestion where vehicles following behind the AV could experience wave smoothing.

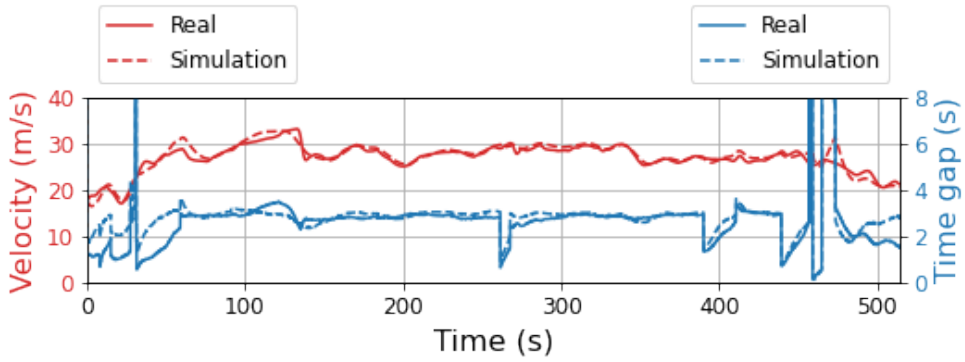


Figure 6.8: Comparison of velocity and time-gap between a real (solid) and simulated (dashed) roll-out. There are small divergences that occur around the cut-ins but the car mostly maintains a three-second time-gap in both cases.

Finally, we analyze the potential fuel efficiency improvements from the validation experiment. The third column in Fig. 6.6 depicts the energy savings obtained when replaying in simulation using the trajectories collected during the experiments using the counterfactual cut-in mechanism mentioned earlier. We observe that the fuel efficiency of the AV has improved by 8% with additional small gains for the IDM vehicles. Fig. 6.9 shows the density of accelerations taken by the IDM vs. the AV; the higher density of large accelerations of the IDM vehicle are likely the reason for the improved fuel efficiency of the RL AV over the IDM AV. Unfortunately, the day of the deployment featured limited congestion so potential improvements are smaller than might be observed in heavier traffic conditions. More experi-

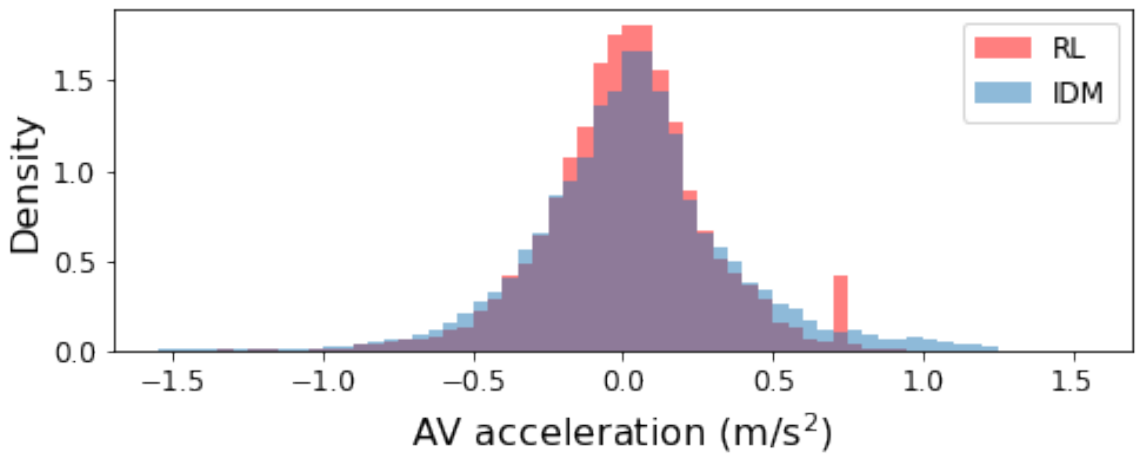


Figure 6.9: The density of the AV acceleration when simulating an AV or an IDM vehicle behind leader trajectories from the tests. The AV case places less mass at high, energy-consuming accelerations. The peak observed at 0.75 corresponds to the lead vehicle being out of range due to cut-outs.

mental testing on a number of days are needed to provide conclusive experimental energy savings results.

6.6 Conclusions and Future Work

In this chapter we propose and test a pipeline that allows for effective validation and training of traffic smoothing controllers. We collect over 700 km of training data that is used to build a controller validation system. This system avoids the fundamental modeling issues that have restricted the learning or design of traffic smoothing controllers to relatively simple settings, or prevented them from deployment on real cars. In our validation system, we use Policy Gradient methods to train a controller that improves the MPG of an AV by 16% and has benefits for the following human vehicles. We then construct a pipeline for porting these controllers to four AVs and perform physical validation experiments over three days. The behavior of the vehicle on the validation experiment closely matches its expected simulation behavior, suggesting that our pipeline is an effective mechanism for validating controllers.

There are a few missing features in our environment that merit further work. First, our simulator lacks counterfactual lane-changes. In future work, this can be addressed using the observed lane changes in the data to build a single-lane lane changing model that can be used to extend our simulation. In terms of the Markov Decision Process we design, our controller is memory-free, which may prevent the agent from learning a predictive model of downstream speeds that can be used for further smoothing. Additionally, we do not penalize

the energy consumption of the platoon; the addition of this penalty may lead to qualitatively different behavior. Finally, additional field experiments can support the assessment of our approaches in a range of traffic congestion levels.

While this work successfully demonstrates that individual controllers can be deployed safely and maintain their intended behavior in highway traffic, it represents only the first step toward large-scale traffic smoothing applications. The pipeline validated here with four autonomous vehicles provides the foundation for scaling to much larger deployments, where system-level coordination and macroscopic traffic effects become relevant. The challenges of coordinating dozens of vehicles, integrating live traffic data, and managing heterogeneous vehicle capabilities require extending these deployment methods to hierarchical control architectures, which forms the central contribution of the following chapter.

Chapter 7

100 RL-controlled AVs Highway Field Experiment

The successful deployment of individual controllers demonstrated in the previous chapter validates the trajectory-based learning approach but represents only a proof of concept for the broader vision of using autonomous vehicles to improve highway traffic flow. This chapter presents the culmination of the research program developed throughout this thesis: the deployment of 100 connected and automated vehicles in a coordinated field experiment on Interstate 24. Moving from 4 to 100 vehicles introduces qualitatively different challenges that require hierarchical control architectures, integration of live traffic data feeds, and coordination across heterogeneous vehicle platforms with varying sensing and actuation capabilities. The MegaController framework developed in this chapter combines centralized speed planning with decentralized vehicle control, incorporating many of the algorithmic innovations from Part I while addressing the practical constraints of large-scale deployment. This experiment represents one of the largest autonomous vehicle traffic smoothing studies conducted to date, providing empirical validation of learning-based traffic control methods at unprecedented scale.

This chapter is based on [131, 132]: Jonathan W. Lee*, Han Wang*, Kathy Jang*, Nathan Lichtlé*, Amaury Hayat*, Matthew Bunting*, *et al.* "Traffic Control via Connected and Automated Vehicles: An Open-Road Field Experiment with 100 CAVs." *IEEE Control Systems Magazine*, 2025.

7.1 Introduction and Related Work

The deployment validation studies presented in the previous chapter demonstrated that individual controllers can successfully bridge the simulation-to-deployment gap and operate safely in highway traffic. However, these small-scale experiments, while encouraging, leave fundamental questions about the potential for autonomous vehicles to create system-wide

benefits when deployed at meaningful scale. This chapter presents the answer to these questions through one of the largest field experiments in learning-based traffic control to date: the coordinated deployment of 100 autonomous vehicles on Interstate 24.

The promise of generating societal benefit from autonomous vehicle technology has long been capturing the imagination of researchers, legislators, and popular culture. Its footprints can be seen in the earliest stages of modern AV research and development in the United States. AV technology advancements, spanning over three decades of work, are sometimes broken down into three generations:

Generation One. In the 1990's, the Federal Highway Administration (FHWA) established the National Automated Highway System Consortium (NAHSC) to demonstrate the potential of an automated vehicle and highway system for societal benefit [133]. The consortium partners included University of California, Berkeley, California Partners for Advanced Transit and Highways (PATH), and General Motors (GM), among several others. In the implementation of Demo '97 on Interstate 15 in San Diego [134], California, the consortium introduced and/or brought to realization concepts such as Adaptive Cruise Control (ACC), Vehicle-to-Vehicle (V2V) Communication, and Cooperative ACC (CACC) [135, 136, 137]. The demonstrations involved high-speed platooning, made possible by V2V and CACC, as well as infrastructure sensor integration. Moreover, it laid the foundation for modern automated highway system AHS architectures, emphasizing the importance of vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications [138]. Some of the key challenges and barriers faced at the time included scalability due to infrastructure, reliable communication systems, and accurate vehicle positioning. Its eventual demise came in the late 1990's with tightening research budgets at USDOT.

Generation Two. In the 2000's, the Defense Advanced Research Projects Agency (DARPA) introduced the DARPA Grand Challenge. In the first Grand Challenge (2004), a $1 million prize was offered to any team whose autonomous vehicle could complete a 150-mile course in the Mojave Desert; no teams were successful. The second Grand Challenge (2005) doubled the prize money, and five teams were successful in completing the course [139]. The third Grand Challenge (2007) moved to an urban environment. The DARPA Grand Challenge competitions were notable for creating a renewed interest in autonomous vehicles and directly supported efforts to commercialize successful technologies [140] in the following decade.

Generation Three. In the 2010's, many major automotive manufacturers (such as GM, BMW, Audi, and Tesla) and technology companies (such as Google/Waymo, Uber, Lyft) began developing autonomous vehicle technology for commercialization purposes. The investments have generated an industrial ecosystem centered on hardware, software, and services designed to advance fully autonomous driving. Today multiple companies are testing AVs in numerous public roads in the US, with Waymo most notably operating an autonomous ride-share service in Phoenix, San Francisco, and Los Angeles without a safety driver in the

vehicle, and Cruise, which operates autonomous shuttles in San Francisco. At the same time, Level 1 and Level 2 automated vehicle technologies, including ACC, are now widely available in the commercial market. These commercial systems predominantly focus on safety and driver comfort [141], rather than improved transportation system efficiency [116].

Many experiments involved using CACC within platooning experiments for improving fuel efficiency on highways [142]. In 2009, the Safe Road Trains for the Environment (SARTRE) [143] three year project was funded by the European Commission to investigate the benefits of platooning (following previous European projects such as PROMOTE CHAUFFEUR I + II or Konvoi), where a leading bus or truck driven by a professional driver controls a following platoon of a few heavy or lights vehicles, and achieving up to 16% energy savings [144] on public highways. In other works, Energy ITS [145] deployed a platoon of three fully-automated trucks in Japan in 2008. In 2011, nine teams developed CACC controllers with V2V communication and compared them in urban and highway settings as part of the Grand Cooperative Driving Challenge (GCDC) [146, 147] held in the Netherlands, and later reiterated in a 2016 edition [148]. Between 2015 and 2017, the California PATH program has been involved in CACC truck platooning experiments on public highways [149]. More recently, the European ENSEMBLE project [150] investigated multi-brand truck platooning to integrate V2V technologies in all European brands of trucks.

Additional studies have experimentally demonstrated the benefits of vehicle platooning. In [151, 152], CACC platoons were developed and deployed on commercially available vehicles using linear control theory and string stability. These CACC platoons were shown to outperform ACC vehicles in terms of disturbance dissipation. [153] showed in simulation that a moderate penetration rate of CACC vehicles can significantly increase network capacity. The research in [154] demonstrated that CACC control allows for safe platooning at low following distances and high speeds, thus improving traffic throughput. Furthermore, [155] indicated that energy consumption in freight trucks could be reduced by driving at low following distances, due to decreased air resistance. Recent experiments in [156] built upon these prior CACC studies by employing model predictive control V2I communication, showing that vehicle platoons can enhance throughput on arterial roadways with traffic lights.

Our work brings full circle the nascent imaginings from the early 1990's. The Generation One promise of automating highways for societal benefit began to manifest in the late 2010's within our team and in several groups around the world. In 2016, a group led by co-authors Delle Monache, Piccoli, Seibold, Sprinkle, and Work replicated the Sugiyama experiment [157] of string instability in a ring road; in what is known as the "Arizona ring experiment," they further demonstrated that phantom jams can be reduced using partially automated vehicle (AV) technologies and specially-designed algorithms [15]. This work was followed shortly by co-author Bayen's lab at UC Berkeley with the development of `Flow`, a software package to interface then-state-of-the-art microsimulation software with deep reinforcement learning (deep-RL) libraries [158, 159, 105, 117, 160, 161]. With `Flow`, the UC Berkeley team was able to independently train an AV controller to replicate the findings of the Arizona ring experiment [162] and generalize them to a variety of other settings, such as freeway and urban traffic [163]. (Note the distinction in terminology between "autonomous vehicle" and

"automated vehicle," specifically because our past and present work focuses on automated longitudinal control as Lagrangian traffic controllers.)

In 2019, co-authors Bayen, Piccoli, Seibold, Sprinkle, and Work united to form the Congestion Impacts Reduction via CAV-in-the-loop Lagrangian Energy Smoothing (CIRCLES) Consortium (see https://circles-consortium.github.io/) [164, 165, 166], joined shortly thereafter by co-authors Lee and Delle Monache, each leading various aspects of the CIRCLES project. The CIRCLES project seeks to extend these prior research efforts to real traffic. To achieve our goals, our group designed a modular, hierarchical control framework, consisting of a centralized Speed Planner and decentralized Vehicle Controllers, and implemented it on 100 vehicles in a large-scale field operational test, dubbed the MegaVanderTest (MVT). Our aim is to deploy flow-smoothing ACC-enabled AVs that don't require explicit communication and are mixed within highway traffic in an unstructured way, contrary to many of the CACC truck platooning experiments previously mentioned. As many modern vehicles come equipped with ACC technology, the focus of this work is to modify the ACC algorithm to become significantly better at flow smoothing, without requiring additional hardware, road infrastructure, or V2V communication capabilities.

This chapter presents our control system design and subsequent analysis of field test results. Diverse candidates for each module of the framework are developed utilizing cross-disciplinary knowledge and tools, including ordinary differential equation/partial differential equation (ODE/PDE)-based flow control [167, 168], deep-RL [132], stabilization theory [169], functional analysis [170], optimal control on microscopic and macroscopic systems [171, 172], approximation theory [173], mean-field limits [174], non-entropic solutions to hyperbolic systems [175], model predictive control (MPC) via linearly constrained quadratic programming (LCQP) [176], kernel smoothing [177], traffic flow theory [178], variable speed limit [179], and many other areas. The control system was then evaluated and tested in the open road as part of the MVT, the largest deployment of AVs designed to smooth traffic flow. In this test, we deployed 100 AVs on Interstate 24 (I-24) near Nashville, TN in November 2022. The experiment coincided with the debut of I-24 MOTION (https://i24motion.org) [180, 181], a four-mile section of I-24 near Nashville, TN to capture ultra-high resolution trajectory data of all vehicles.

The rest of the chapter is organized as follows: Section Section 7.2 introduces the MegaController, our controller architecture which consists of several components that operate together. Section Section 7.3 then details the design and functioning of each of these individual components. Section Section 7.4 explains how controllers are selected for the test, and finally Section Section 7.5 introduces the experimental design for the 100 AV deployment, as well as implementation details for the selected controllers, deployment on hardware and data collection procedures. Furthermore, several sidebars are spread throughout the chapter, which can serve different roles: giving background about certain parts of the projects, diving into much more details about certain aspects of the project, or presenting interesting work that happened before or in parallel to the project, or that didn't make it into the final deployment.

7.2 Design of the Controller Architecture

We introduce the **MegaController**, a control framework, depicted in Figure 7.1, for the mixed autonomy traffic flow problem. Mixed autonomy refers to the setting in which some vehicles are automated, while other vehicles are manually controlled (level 0 automation). The primary design paradigm of the proposed control framework is to achieve two goals: hierarchy for task allocation and modularity for control flexibility.

We designed a hierarchical structure to efficiently coordinate the control goals between macroscopic traffic flow optimization [182] and microscopic vehicle control [183, 184, 185, 186] and to efficiently solve the computational task allocation problem between the server and vehicle sides. There is an inherent interplay between the two components, as the Speed Planner informs the Vehicle Controller of downstream events, and the vehicles are instrumented to report observations to the server for data aggregation.

7.2.1 Use of modular design

A modular structure is used for two important purposes: (1) to facilitate a diversity of controller design approaches, and (2) to enable a heterogeneous vehicle fleet with different sensing and actuation capabilities.

As long as each controller utilizes the available interfaces, it is technically straightforward to exchange different controllers when using a modular design. This decision allows us to support a wide range of expertise from our team who may explore many different designs, each with a different technical focus or approach. While there may be subtleties for stability, convergence, and other properties if integration decisions are made strictly through type matching, our modular approach mitigates a key challenge: the dynamics of the roadway and our ability to control it may not be fully understood until testing begins. Since testing may only be possible at a scale when we can evaluate its impact, we will be very close to the full test deadline when we make final decisions. By embracing a modular approach, we can use data from tests that concluded less than 24 hours earlier to decide what controller to run during the following test.

The fleet of vehicles used in the MVT represented 3 different years, makes, and models. The fleet was composed of 2023 Nissan Rogue, 2022 Cadillac XT5, and 2020-2021 Toyota Rav4. Trade secrets and design differences between the systems on these vehicles mean that it is not possible to have a uniform system interface for sensing and control. For example, although each car we used has adaptive cruise control which uses forward facing radar for safety, the data from those sensors were not available to our system for all vehicle types. Designing a controller that can work on only one vehicle type may give greater controllability, but reduces the potential impact of the design when deployed at a societal scale.

Thus it is a design strength that our system can operate across vehicles with myriad different sensing and control modalities. While the interfaces to the Speed Planner remain the same, different vehicles will have different implementations that have completely unique characteristics compared to other vehicles, similar to textbook design patterns in software

engineering for producer/consumer architectures. This allows a flexibility to adapt to a heterogeneous fleet, theoretically opening the door to many other vehicle configurations and constraints.

In the MegaController design, there are two distinct components:

- A server-side **Speed Planner**, which is a centralized planner unit that provides high-level macroscopic speed suggestions based on periodic state updates from the distributed vehicles and external macroscopic data source, like INRIX employed in this study. The algorithms deployed on the server-side are designed to handle computationally-heavy data aggregation and macro-state tasks.

- A vehicle-side **Vehicle Controller**, which is a networked decentralized controller [187] that commands local actuation of the vehicle. The algorithms deployed here take into consideration the target speed suggested by the server-side planner unit, the latency of that information, and any observations from the vehicle's onboard sensors.

Borrowing conceptually from object-oriented programming, the interfaces are agreed upon *a priori*, allowing an abstraction of individual components for ease of design and testing.

7.2.2 Speed Planner

The objective of the Speed Planner is to optimize the overall traffic flow efficiency by providing macroscopic guiding speeds for AV fleets using realtime traffic data from multiple heterogeneous sources. While optimality for our deployment was defined by flow efficiency, our framework allows for arbitrary objective functions.

The data sources at our disposal are the set of recent vehicle ping messages from our AV fleet using our custom hardware and software stack (detailed later in Figure 7.8), as well as vehicular velocity data within distinct sections of highway through INRIX, a provider of live data for traffic, parking, and other transportation data. INRIX data has been studied in the literature for its applicability in traffic estimation, and validated by researchers for its accuracy using *post hoc* methods [188]. Utilizing these heterogeneous data sources presented some key challenges and considerations, namely unknown latency of external data sources and differences in spatial and temporal resolution of those data sources when fused with our vehicle message data. Although the specific aggregation algorithm is not publicly known, INRIX data latency is explored in [189], which describes delays between 3-5 minutes, and in some cases up to 10 minutes. A summary of data specifications can be found in Table 7.1.

Targeting the challenges, the two main functions of the Speed Planner are (1) Traffic State Estimation (TSE) enhancement and (2) the design of the Speed Plan, an ideal target speed profile. The TSE enhancement module is designed to eliminate the effects of inherent latency in the third-party data source and to improve the spatial and temporal resolution of the input traffic data. The target design module generates Speed Plans with the goal of reducing vehicle energy consumption and increasing the overall throughput of the traffic flow. Various

Data Source:	INRIX	AV Ping
Period (seconds)	60	1
Latency (seconds)	∼180	negligible
Segment Length (miles)	∼0.5 (varied)	N/A
Lateral aggregation	Lane averaged	Lane specific

Table 7.1: A summary table of the data sources for traffic state estimation (and subsequent processes). INRIX aggregates data from a fleet of vehicles on the road. The AV Ping data come from our own fleet, where each car posts speeds to the server through an API that contains speed (from CAN) and positioning (from GPS), as well as timing and vehicle identity information.

approaches are utilized to develop the target design module, including kernel smoothing [177], ODE/PDE-based flow control [167] and vehicle trajectory optimization [183].

7.2.3 Vehicle Controller

The Vehicle Controller enables each car to use traffic information provided by the Speed Planner in context with local state, such as the distance to the vehicle in front, relative speed, etc. The Vehicle Controller aims to improve the flow of traffic while ensuring the safety of both the AV and other vehicles around it.

Input data for the Vehicle Controller are obtained from the in-vehicle network, and may include other inputs such as the target speeds from the Speed Planner. A subset of data from the CAN bus are made available to the vehicle controller, and vary depending on the vehicle make/model/year. Realizing control algorithms on vehicles required new computer hardware, electronics, and software development [122, 125, 107, 190, 191, 192] because of the heterogeneity in vehicle data and control interfaces.

Depending on the vehicle platform, we have explored options including speed-based control [193, 127], acceleration-based control [107, 92], and ACC-based control [132]. Speed-based control was initially explored, but then deprecated as acceleration based control best matched desired goals from algorithm designers. The acceleration-based and ACC-based approaches fit into the overall architectural design. Torque-based control (that is, applying the requisite torque directly to the gears) would be the most prescriptive control with tightest actuation, but it also requires a level of CAN bus access that prevents quick return of the vehicle to stock state. The majority of the vehicles used in the experiment were stock vehicles that were later returned to the manufacturer, so all changes to enable experimental control needed to be minimal and non-invasive changes to the vehicle. Therefore, for this deployment, our architecture included two types of control capabilities on our heterogeneous fleet, which are

summarized in Table 7.2.

Vehicle(s):		Toyota RAV4	Nissan Rogue, Cadillac XT5
Actuation Type:		Acceleration	ACC
Sensing	Ego Position	✓	✓
	Ego Speed	✓	✓
	Ego ACC Speed Setting	✓	✓
	Ego ACC Gap Setting	✓	✓
	Leader Space Gap	✓	✗
	Leader Minicar	✓	✓
	Leader Relative Speed	✓	✗

Table 7.2: A summary table of the sensing and actuation available to our system, for the heterogeneous vehicle fleet. Space gap refers to the distance between the ego vehicle's front bumper and the leader vehicle's rear bumper. The minicar refers to a boolean indication whether the ego vehicle's sensors detect the presence of a leader vehicle (an approximate 80-100 meter maximum distance). Relative speed is the leader vehicle's speed minus the ego vehicle's speed.

7.2.3.1 Adaptive Cruise Control Modeling

Adaptive Cruise Control (ACC) is an advanced driver assistance system that automatically operates vehicles at a set driving speed or keeps a set driving gap with respect to a leading vehicle. Operation of ACC requires a speed setting and a gap setting input by the driver. When there is no leading vehicle, ACC automatically adjusts the vehicle's driving speed to the speed setting preset by the driver. When there is a leading vehicle ahead, the system automatically regulates the gap with respect to the leading vehicle to a separation set by the driver, while not driving faster than the speed setting.

An ACC-controlled car therefore be modeled as a dynamical system composed of two control modes: speed-control mode and gap-control mode. The system switches between two modes depending on the proximity and speed of a leading vehicle. When there is no nearby leading vehicle, the system is in speed-control mode, in which the system tracks the speed setting. On the other hand, when the system is in gap-control mode, the system tracks both the leading vehicle speed and the gap setting.

To model an ACC-controlled car, we used parametric models for each control mode.

Mathematically, the speed-control mode can be written as

$$a_e = f_p(v_e, v_{\text{ref}}) \ , \tag{7.1}$$

where a_e is the acceleration of the ego vehicle, v_e is the speed of the ego vehicle, and v_{ref} is the speed setting. $f_p : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a parametric model of parameters p.

Gap-control mode can be written as

$$a_e = f_q(v_e, v_l, d_{el}, g_{\text{ref}}) \ , \tag{7.2}$$

where v_l is speed of leading vehicle, d_{el} is the space gap between the ego vehicle and the leading vehicle, and g_{ref} is the gap setting. $f_q : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a parametric model of parameters q.

To fit parameters p and q for speed-control mode and gap-control mode, field experiments are carried out using a Nissan Leaf vehicle equipped for data collection. For the speed-control mode, data collection is done with a variety of initial speeds and speed settings so that dynamic responses with different speed changes can be collected. For the gap-control mode, highway driving data under different traffic conditions is collected so that dynamic response over a range of speeds can be covered in the data set. The collected data is firstly smoothed to reduce sensor noise. Some outliers of abnormal driving conditions (for example, temporally near cut-in and cut-out events) are removed before fitting the models. The model is validated by comparing the dynamic response of the model in simulation to the response of the real system in field experiments. While the models are fitted for the Nissan Leaf, they are assumed to be acceptable approximations for the ACC systems of other vehicles in our AV fleet.

7.2.3.2 Trajectory Simulator

Establishing a robust, representative, and reliable simulation environment is key for the successful implementation of our wave-smoothing controllers in autonomous vehicles, as it ensures their efficient operation across a wide range of traffic conditions. Our simulator [107] is based on real human driver data collected on the highway[123], and has been extensively used for designing, training and evaluating the different types of controls implemented in this work, assessing their safety, energy-reducing performances, robustness and smoothness. This sidebar describes the procedure by which highway trajectory data was gathered and a simulator created from it.

Acquisition of Data The trajectory dataset [123] for our study were recorded on a 14.5-kilometer segment of I-24, located southeast of Nashville, Tennessee. An example recorded speed trajectory profile can be seen in Figure 7.2. An instrumented vehicle is used to gather data which logs vehicle controller area network (CAN) data through `libpanda` [122] and GPS information from an in-built receiver. The CAN data collection comprises measurements like the speed of the vehicle under consideration (ego vehicle), the relative speed of the lead vehicle (the vehicle in front), the instantaneous acceleration, and the space gap (distance from bumper to bumper).

Refining Raw Data Each drive's raw data is stored in two separate files: a CAN data file and a GPS file. The relevant data are extracted from the CAN file and adjusted to match the GPS time, which is recorded at 10 Hz. The high-frequency CAN data are downsampled and interpolated linearly to align with GPS time, whereas low-frequency CAN data are subjected to linear interpolation to match the 10 Hz GPS time. GPS position data is used to calculate the distance traveled and direction. As westbound data usually demonstrate more consistent congestion, they are primarily used for training, comprising 60 trajectories, which translates to 8.8 hours and 772.3 kilometers of driving.

Analyzing the Dataset The trajectory dataset encapsulates a wide variety of traffic conditions ranging from nearly stationary congested traffic to maximum speed free-flow traffic, including diverse acceleration and deceleration patterns associated with stop-and-go traffic. The example trajectory displayed in Figure 7.2 demonstrates the ego vehicle's quick transitions between low and high speeds.

While the primary interest lies in mitigating high-frequency waves that are common in congestion, Figure 7.3 indicates the tendency of speeds in the training dataset towards the higher end. Despite the possibility of simplifying the learning problem by filtering out high-speed data, congestion zones are often immediately followed by high-speed areas. To ensure our controller's competent behavior at high speeds and during transitions from high to low-speed zones, the training dataset retains both low and high-speed data.

Car-Following Models *Car-following models* (CFMs) [194, 178] are systems of ordinary differential equations (ODEs) describing the dynamics of each vehicle on the road, where drivers react to the changes in the relative positions of the vehicle ahead. The ODEs could describe the vehicle velocity only and those are first-order models, or the velocity and the acceleration and those are second-order models. Second-order CFMs are of the form

$$\dot{v}(t) = f(s(t), v(t), \Delta v(t)) \ , \tag{7.3}$$

where s is the gap to the vehicle ahead measured in meters, v is the velocity of the vehicle, commonly measured in m/s, and Δv is the velocity difference or the approaching rate to the vehicle ahead, commonly measured in m/s. An elegant CFM is the Optimal Velocity Model (OVM) [195]

$$f(s, v, \Delta v)_{\text{OVM}} = \alpha \left[V(s) - v \right] \ , \tag{7.4}$$

where the optimal velocity function, $V(s)$, determined by the gap s to the vehicle ahead, is a positive monotonically increasing function, with $s \to \infty$ asymptote at the speed limit. Some variations of this model were proposed to avoid car collisions, an example is the Optimal Velocity Follow the Leader Model (OVM-FtL):

$$f(s, v, \Delta v)_{\text{OVM-FtL}} = \alpha \left[V(s) - v \right] + \beta \left[\frac{\Delta v}{s^{\nu}} \right] \ , \tag{7.5}$$

where ν is a positive exponent, and β is a positive braking coefficient (measured in $\mathrm{m}^{\nu}/\mathrm{s}$). Another example of second-order CFMs is the Intelligent Driver Model (IDM), introduced in [196] and suitably adapted and used in this work. The IDM acceleration function is

$$f(s, v, \Delta v)_{\mathrm{IDM}} = a \left[1 - \left(\frac{v}{v_0} \right)^{\delta} - \left(\frac{s^*(v, \Delta v)}{s} \right)^2 \right] , \tag{7.6}$$

where $s^*(v, \Delta v) = s_0 + vT + \frac{\max\{0, v\Delta v\}}{2\sqrt{ab}}$. Here v_0 is the desired velocity on an empty road (measured in m/s), s_0 represents the minimum spacing between vehicles (measured in m), T is the minimum possible time gap to reach the vehicle ahead (measured in s), δ is an acceleration exponent, and a and b are the maximum vehicle acceleration and minimum desired comfortable deceleration, respectively (measured in $\mathrm{m/s}^2$).

Developing the Training and Evaluation Framework To exploit the gathered data, a single-lane training environment is designed where an AV follows the trajectory data recorded from human drivers. The human driver is simulated at the front of a vehicle platoon, followed by the AV, and then a number of vehicles operating according to the IDM. This setup guarantees the growth of waves in congestion, due to the string-unstable nature of the chosen IDM parameters. While a comprehensive micro-simulation of I-24 might allow training with more complex vehicle interactions, the proposed simulator focuses on realistic driving dynamics representative of the highway's wave types and drivers' reactions to wave formation. The simulator's efficiency is also commendable as it achieves 2000 steps per second, whereas a micro-simulation of the complete 14-kilometer section would be computationally expensive due to the thousands of vehicles in congestion.

7.2.3.3 Acceleration-based Controller

While acceleration controllers don't have natural safety enforcement such as the one speed controllers have, we have found a way to add safety enforcements. Using new approaches in [197], we demonstrated that it was possible to constrain unsafe accelerations when they were passed to the vehicle. This opened the door to use of acceleration-based control on cars that support it. Acceleration-based control is a natural analog to how most of us drive: we press down on the accelerator pedal when we want to go faster, and we either let off the accelerator, or press the brake pedal, when we want to go slower.

When making control requests to the car, we provide a desired input over the vehicle's CAN, which is interpreted by the powertrain and ACC subsystems of the car. This desired input is not strictly a direct acceleration command: it is processed by the vehicle's ACC system which interprets the commanded acceleration and decides whether it will meet it (a) by actuating the brake, (b) by decreasing acceleration through reducing the throttle, (c) by keeping the throttle constant, or (d) by increasing the throttle. Examples of how these dynamics from step inputs have been observed can be found in [193]. The specific input selected by the car is a function of many different vehicle dynamics and perhaps even trade

secrets for reducing wear-and-tear of parts, so there is no single mapping or predictive function that easily codifies the transfer function of the system once an acceleration is requested.

Using acceleration-based control *implicitly* adds safety challenges, as the controller must ignore all existing acceleration commands from the car—including any acceleration commands that come from the safety subsystems on board. Safety wrappers such as [197] are needed to prevent rear-end collisions, and additional care must be taken to evaluate external sensors. Further, environmental challenges such as lane-changes by other vehicles require our system to ensure that our dynamics do not inadvertently cause their own stop-and-go waves. We decompose the behavior of this controller into two modes:

- A **base controller** prescribes a performance-based desired acceleration under "ideal" conditions. The base controller is designed to output an acceleration that satisfies a specific performance-based goal, for example, reducing energy footprint or minimizing the magnitude of accelerations. In general, care is taken during its design-phase to ensure that other aspects of the traffic flow are not negatively impacted. Note that the base controller should still, by design, be safe and collision-free, under the assumption that accelerations are actuated perfectly.

- A **lane-change recovery controller** modulates the base controller's desired acceleration in the event of discontinuities in the lead-vehicle space gap. The acceleration output of the base controller can change dramatically if there is a time-discontinuity in the input signals. Most commonly, this appears in the form of a lane-change by another vehicle in front of the ego vehicle. Since a lane-change is out of the ego vehicle's control, it can theoretically put the ego vehicle into an unsafe state if the objective were to control to a time gap with discontinuous input. This mode recognizes that discrete changes in the space gap are not controller failures, so the objective of this controller is to comfortably return the vehicle to a safe state where the base controller can resume actuation.

7.2.3.4 ACC-based Controller

Another option for controlling vehicles is to update vehicle's *Adaptive Cruise Control* (ACC) set points electronically. This mimics what the driver can do through buttons on the steering wheel, and has the benefit of keeping the safety features of the car's ACC systems in the loop. This approach is contrasted with the acceleration-based approach in a few ways, notably:

- The controllability of the system is affected and the bandwidth of the controller is likely reduced, since the stock ACC has its own gains and modes.

- The multi-model design features of the acceleration-based approach (namely the lane-change recovery controller and safety wrapper) are not needed here, as those features are built into the stock ACC algorithm.

To understand the changes in controllability, we explored the vehicle's ACC dynamics with a goal to make design decisions on whether rate limits or other constraints on the input signals would be required for stability or performance reasons. Every vehicle make and model equipped with ACC has its own unique mapping of state space (for example, ego speed, space gap, leader speed, etc.) and ACC inputs (speed setting and gap setting) to actuated acceleration. Architecturally, we take as given that this model is available for the design of the base ACC controller. For a more detailed discussion of ACC and our specific approach to obtaining a model for our algorithm design, see Section 7.2.3.1.

As a result, the only component of the ACC-based controller that requires specific designing is the base controller. Just as with the acceleration-based controller, the base controller is performance-based, striving for some performance goal. Similar design philosophies are applied as before. Uniquely different, as noted above, is that this base controller does not need to explicitly consider safety and vehicle-specific dynamics—the controller design can implicitly account for the vehicle-specific dynamics by utilizing the ACC model in a feedback loop.

7.3 Design of Controller Components

Here, we describe in detail how each of the modular components was designed. The discussion is in a parallel structure to the previous section, with particular focus on controllers that were actually implemented for deployment. In the lifespan of the project, we explored many approaches for designing the controller.

7.3.1 Speed Planner

Based on the hierarchical framework, Figure 7.4 indicates the implementation of the speed planner we tested in the MVT. As introduced in Table 7.1, INRIX and probe vehicle data are posted to the central database in different frequencies: each vehicle makes its post approximately every 1 second, and a server-side process inserts new data from INRIX approximately every 60 seconds. It is important to note that while the INRIX data provides a single speed across all lanes, the AV pings provide lane-level speed information.

The sequence of events of a Speed Plan publication can be summarized as follows:

1. Each new INRIX update is combined with historical INRIX data, and fed into the prediction module.

2. All vehicle observations from the previous 60 seconds are fetched, and fused with the INRIX prediction, to obtain a lane-level traffic state estimate.

3. The lane-level TSE is smoothed with the forward-kernel average.

4. Bottleneck identification is performed with the smoothed lane-level TSE.

5. If a standing bottleneck is identified in the lane, a deceleration region is prescribed as a buffer segment. For all other regions, the smoothed lane-level TSE is used as the lane-level Speed Plan.

6. Publish the Speed Plan for all lanes.

The subsequent sections delve deeper into the intricacies of the TSE Enhancement and Target Design, providing a comprehensive understanding of the steps outlined above. Wang et al. [167] provides in-depth methodology description of the Speed Planner.

7.3.1.1 TSE Enhancement

The TSE enhancement module comprises two main components: the INRIX prediction module and the data fusion module. The prediction module's primary role is to minimize the latency issues associated with INRIX realtime data, especially when applied to vehicular control, as opposed to its standard use for general traffic insights. The fusion module complements this by integrating realtime data from our system's probe vehicles, enabling a more detailed, lane-level TSE with enhanced time-space precision. A detailed mathematical explanation of the TSE enhancement module for MVT implementation is available in [167].

To achieve a lane-level TSE with superior spatial detail, we further segment the INRIX data into smaller units. After generating the INRIX prediction, it is merged with realtime data from the system's controlled vehicles. Given that each AV communicates with the server at a rate of 1Hz, we receive 60 ping records from each vehicle for every Speed Plan generation. These records help determine the average speed of each vehicle over the previous update period, which then updates the TSE for the respective sub-segment. Assuming that our drivers consistently stick to their designated lanes (a behavior largely confirmed through systematic data review), we can generate lane-specific TSE estimates by combining vehicle data with the broader INRIX speed data. When it comes to data fusion, we give precedence to data from our vehicles over INRIX data, considering the inherent characteristics of both sources. Our vehicles gather and relay perception data through a system that is both observable and controllable, with quantifiable error and latency as detailed in [198]. In contrast, the INRIX realtime API employs averaging techniques to forecast over a given period. This method might introduce inaccuracies at specific points within that timeframe, which could impact our control execution. The INRIX system, in essence, operates as a somewhat opaque system, with its error and latency aspects largely inferred from provider descriptions. In this chapter, we adopt the following notation to represent the discrete TSE:

$$\{(x_j, \bar{v_j}), j \in \mathcal{J}\}, \tag{7.7}$$

where j represents the index of road segments, x_j is the postmile coordination of the central of road segment j and $\bar{v_j}$ is the average speed of the corresponding road segment j. Wang et al. [167] details the procedure of data fusion applied in MVT.

7.3.1.2 Target Design

This section introduces the Target Design's main modules, namely the kernel-based smoothing and the learning-based buffer design. The kernel smoothing processes the enhanced TSE at each time step using a chosen kernel to improve the fuel consumption caused by the shockwave in a high-density traffic flow. The buffer design utilizes Reinforcement Learning (RL) to form a buffer area upstream of the standing bottleneck with the goal of improving throughput at the bottleneck. The target speed suggested by the RL is employed in a mathematical model of traffic, represented by a strongly coupled Partial and Ordinary Differential Equation (PDE-ODE). The outcome of this mathematical model is an identification of traffic density $(t, x) \mapsto \rho(t, x)$. The kernel smoothing then receives this information for learning the velocity of the next time step.

In kernels smooth module, we rely on enhanced TSE data to synchronize the driving speeds of automated vehicles. In particular, vehicles are assigned target speed profiles contingent on traffic state information, which is shared and common among all AVs.

At any fixed time step t, the desired speed profile $\mathbf{v} : \mathbb{R}_+ \times \mathbb{R} \to \mathbb{R}_+$ is extracted from enhanced TSE utilizing kernel methods. First, we preprocess the sparse TSE data by interpolating the discrete data pairs $(x_i, \bar{v}_i)$ to a continuous speed profile $(t, x) \in \mathbb{R}_+ \times \mathbb{R} \mapsto v(t, x)$, as an approximation of the average speed of a higher granularity traffic at a position x and at time t. Then, for any fixed time $t = t_\circ$ we obtain the desired speed by applying a kernel function $K(\cdot)$ at a position $x = x_\alpha$:

$$\mathbf{v}(t_\circ, x_\alpha) = \frac{\int_{x=x_\alpha}^{x_\alpha+w} K(x_\alpha, x) v(t_\circ, x) dx}{\int_{x=x_\alpha}^{x_\alpha+w} K(x_\alpha, x) dx}, \tag{7.8}$$

where w is the width of the estimation window. Many different kernel functions, such as Gaussian kernel, Triangular kernel, Quartic kernel, Uniform kernel etc., can be chosen. For the purposes of this chapter, we consider a uniform kernel, the simplest of such mapping. The desired speed profile at a position x_α is accordingly defined as:

$$\mathbf{v}(t_\circ, x_\alpha) = \frac{\int_{x=x_\alpha}^{x_\alpha+w} v(t_\circ, x) dx}{w}. \tag{7.9}$$

When human drivers observe a gap between their vehicle and the one preceding, they tend to accelerate to close the distance. Our proposed desired speed profile aims to slow down *in advance*, although not excessively, to create a gap from the preceding vehicle. This approach takes into account the information provided by the TSE, which indicates the presence of congestion in the nearby downstream area. The proposed desired speed profile is adaptive to traffic states and offers relative robustness, as it only requires one parameter, w, to tune.

For buffer design, we consider the interval $\mathcal{I} \subset \mathbb{R}$ as the region of interest. In addition, we consider a subregion $\mathcal{I}_c \subset \mathcal{I}$ as a congested area. The idea is to determine the controlled vehicle target speed at a time step $t_\circ$, denoted by $\mathbf{u}(t_\circ, x)$, such that the density $\rho(t, x)$ for $x \in \mathcal{I}_c$ and $t \geq t_\circ$ is distributed uniformly through the region $\mathcal{I}$. Determining the

controlled vehicle's target speed will be done in the following steps: (i) Designing a target speed $\mathbf{u}(t_\circ, x)$, given the input $\mathbf{v}(t_\circ, x)$ from kernel smoothing step, (ii) identifying the density $(t, x) \in \mathbb{R}_+ \times \mathcal{I} \mapsto \rho(t, x)$, given the target speed of the controlled vehicle, employing a strongly coupled PDE-ODE model of traffic flow, (iii) Evaluation step in which using the density, the speed profile will be updated by smoothing kernel.

Paper [167] provides details of the procedure introduced above.

7.3.2 Vehicle Controller

Recall in the architectural design of the controller, the Vehicle Controller layer features two distinct methods for actuating the vehicle: acceleration-based control and ACC-based control. Here, we will discuss component implementations prepared for both options.

7.3.2.1 Acceleration-Based Controller: Base Control

The acceleration-based controller requires a baseline controller to use under normal and emergency operating conditions, and a lane-change recovery controller to behave smoothly when cut-ins and cut-outs take place.

The base controller is an explicit, mathematically-defined controller designed to reach and keep an ideal target speed without being trapped in the stop-and-go wave. These two goals are somewhat antagonistic: a naive approach consisting in following the ideal target speed (unless it is unsafe to do so) may result in braking at the same time and with a comparable amplitude as the leading car when caught in the wave.

To address this, the base controller has an anticipation mechanism based on a paradigm that can be summarized as "act swiftly but slightly," which reduces variability in acceleration. It is comprised of three key components: *target*, *anticipation*, and *safety*. The target component infers an ideal target speed for each specific AV based on inputs from the Speed Planner and from the local data available. The anticipation component is an MPC module that aims to anticipate the leader's behavior based on its current acceleration. This enhances the controller's ability to follow the target speed effectively. The safety component is a safety module, that has priority over all the other components and ensures that the AV remains safe at all times. Since car-following models never represent traffic perfectly (see for instance [199, 200]), it is important that the safety module, at least, is independent of the mathematical model used to model vehicle behavior.

The mathematical expression of the commanded acceleration is given by

$$a_{\text{cmd}} = \min(a_{\text{safe}}, a_{\text{target}}, a_{\text{MPC}}) \ , \tag{7.10}$$

where a_{safe}, a_{target}, and a_{MPC} correspond to the *safety*, *target*, and the MPC-based *anticipation*

components. The safety component is given by

$$\begin{aligned} a_{\text{safe}}(t) &= -k(v(t) - v_{\text{safe}}(t)) + \dot{v}_{\text{safe}}(t) \ , \\ v_{\text{safe}}(t) &= \sqrt{2|a_{\min}| \left(h(t) - s_0 + \frac{1}{2} \frac{v_{lead}^2(t)}{|a_{l,\min}|} \right)} \ , \end{aligned} \tag{7.11}$$

where k is a positive parameter, v is the velocity of the ego vehicle, s_0 is safety distance, h is the space gap, v_{lead} is the leader vehicle velocity and $a_{l,\min}$ is the minimal possible leader vehicle acceleration (so maximal deceleration), and $a_{\min}$ the minimal possible ego acceleration.

The target component is defined by

$$a_{\text{target}}(t) = -k(v(t) - v_{\text{target}}(t)) \ , \tag{7.12}$$

where v_{target} is an inferred ideal speed obtained either from the speed planner when available or by integrating local measurements.

The MPC-based anticipation component a_{MPC} is given by

$$a_{\text{MPC}}(t) = \begin{cases} a_{\text{min. brake}}(h(t), v(t), v_{lead}(t), a_{lead}(t)), \\ \qquad\qquad \text{if } P_1 > 0, \ a_{lead}(t) < 0 \\ a_{lead} v(t)/v_{lead}(t), \\ \qquad\qquad \text{if } P_1 \leq 0 \text{ and } P_2 \geq 0, \ a_{lead}(t) < 0 \\ a_{lead} - \dfrac{(v(t) - v_{lead}(t))^2}{2(h(t) - s_0)}, \\ \qquad\qquad \text{if } P_1 \leq 0 \text{ and } P_2 < 0, \ a_{lead}(t) < 0 \\ a_{lead} - \dfrac{(v(t) - v_{lead}(t))^2}{2(h(t) - s_0)}, \\ \qquad\qquad \text{if } P_2 < 0, \ a_{lead}(t) \geq 0, \\ \min(a_{\max}, a_{lead}(t)(1 + k_2(v_{\text{lead}}(t) - v(t))), \\ \qquad\qquad \text{otherwise.} \end{cases} \tag{7.13}$$

where, as in (7.11), h is the space gap and s_0 the safety distance, and k_2 is a positive constant, a_{lead} is the leader vehicle acceleration, $a_{\max}$ is the maximal possible acceleration of the ego vehicle. The terms $a_{\text{min. brake}}$, P_1, and P_2 are given by

$$\begin{aligned} &a_{\text{min. brake}}(h(t), v(t), v_{lead}(t), a_{lead}(t)) \\ &\quad = -\left(h(t) - s_0 + \frac{1}{2} \frac{v_{lead}^2(t)}{-a_{lead}(t)} \right)^{-1} \frac{(v(t))^2}{2} \ , \\ P_1 &= a_{\text{min. brake}}(h(t), v(t), v_{lead}(t), a_{lead}(t)) \\ &\quad - a_{lead}(t) v(t)/v_{lead}(t) \ , \\ P_2 &= v_{lead}(t) - v(t) \ . \end{aligned} \tag{7.14}$$

Further details of this controller can be found in [201].

7.3.2.2 Acceleration-Based Controller: Lane-Change

The design of Lane-Change Recovery Controller relies on global traffic information as well as local traffic states, such as the space gap to the leader vehicle and the relative velocity. The event of lane changing of the leader vehicle creates a discontinuity in the observed local state. Such discontinuities can translate to sudden, large jumps in the controller output, causing large jerk values. This sudden jump in the controller output might be necessary to avoid collision (for example, if a vehicle cuts-in in front of the AV and has low relative velocity). However, in many cases, these jumps are byproducts of the discontinuity in the observation and can be avoided without creating additional safety threats. To remedy this effect of lane-changes in such conditions, we designed a simple lane-change handling mechanism that treats the main acceleration-based controller as a black box and makes minimal assumptions about it. Essentially, the mechanism works by detecting the event of lane changing, assessing the safety conditions created by the event, and (when appropriate) smoothing the output of the controller.

We assume discrete observations with fixed frequency. At a step k, we denote the ego velocity v_k, the relative velocity Δv_k, the space gap s_k, the output of the main controller a_k, and the actual acceleration input to the vehicle u_k. If the lane-change mechanism is not active, we have $u_k = a_k$.

In order to design this mechanism, we make a few natural assumptions about the behavior of the underlying controller. We assume that the controller is a continuous and non-decreasing function of the leader's states (that is, the space gap s_k and relative velocity Δv_k). We also assume that the main controller encodes its own safety measures as the lane-change handling mechanism does not provide any additional safety measures. Beyond these assumptions, the exact form of the main controller is treated as a black box.

We impose multiple criteria for the lane-change handling mechanism to take effect:

1. a lane-change event is detected—this is done by detecting a discontinuity in the space gap to the leader vehicle;

2. the lane-change event is significant—this is measured by the amount of jerk it causes;

3. the lane-change event does not cause safety threats—this is measured by the *time to collision* (TTC) (if $\Delta v_k < 0$) or time gap (if $\Delta v_k \geq 0$) at the lane-change event.

When the lane-change mechanism is activated, it smooths the acceleration by taking a convex combination of the main controller output at the current step and the actual acceleration command in the previous step:

$$u_k = a_k u_{k-1} + (1 - a_k) a_k \ . \tag{7.15}$$

Here $0 \leq a_k < 1$ is a smoothing factor where larger values mean smoother acceleration change. Here we consider a time-varying a_k whose value is a non-decreasing function of the relative velocity and time headway at step k. The rationale for this modeling choice is to make

the smoothing effect stronger the less critical the situation is (that is, larger space gap and relative velocity). The smoothing factor has the following functional form:

$$a_k = c \cdot \left(f_1 \left(\frac{s_k}{v_k} \right) \right) + (1 - c) \cdot f_2(\Delta v_k, s_k) \ . \tag{7.16}$$

It depends on the time headway through the function

$$f_1 \left(\frac{s_k}{v_k} \right) = \tanh \left(t^\star \cdot \frac{s_k}{v_k} \right) \tag{7.17}$$

where the parameter $t^\star$ is chosen such then the function is close to 1 when the time headway $\frac{s_k}{\Delta v_k} \geq h_{\text{safe}}$, where h_{safe} is an acceptable safety time headway. The smoothing factor depends on the relative velocity through the function

$$f_2(\Delta v_k, s_k) = \begin{cases} \frac{1}{2} \tanh \left(\Delta v^\star \cdot \frac{s_k}{|\Delta v_k|} \right) + \frac{1}{2} & \Delta v_k < 0 \\ \frac{1}{2} \tanh \left(\Delta v_k \right) + \frac{1}{2} & \Delta v_k \geq 0 \end{cases} \tag{7.18}$$

where the parameter $\Delta v^\star$ is selected such that the function value is close to 0 when the time to collision $\frac{s_k}{|\Delta v_k|} \leq C_{\text{safe}}$, where C_{safe} is an acceptable safety time to collision. Concretely, we use $h_{\text{safe}} = 2s$, $C_{\text{safe}} = 4.5s$, $t^\star = 1.32$ and $\Delta v^\star = 10.3$. The parameters $c = 0.75$ is chosen by testing on multiple trajectories and lane-change scenarios.

Finally, the lane-change mechanism is deactivated when the main controller acceleration becomes close enough to the output acceleration

$$|a_k - u_k| \leq \epsilon \ . \tag{7.19}$$

An example of this mechanism is depicted in Figure 7.5. In the figure, the dashed red line indicates the lane-change event (cut-in) happening at around 14 seconds. The acceleration profile of the main controller (shown in gray in the bottom left panel) decelerates heavily in reaction to the lane-change. This event is detected and considered safe by the lane-change recovery controller which remained active for 2.9 seconds during which it smoothed the acceleration profile significantly and removed the unnecessary jerk cause by the lane-change.

7.3.2.3 ACC-Based Controller

The ACC-based controller is the version of the controller that was ultimately deployed on 97 of the 100 vehicles during the final MVT. The crucial distinction between the ACC-based controller and the acceleration-based controller introduced in the prior section is the controller's output. The ACC-based controller, rather than providing an acceleration to actuate, provides outputs setpoints for the AV's native ACC system, which controls the AV's longitudinal movements based on those setpoints. We use the stock ACC system's safety assurances and lane-change handling. Thus while this provides a less direct form of control, it is more robust in ensuring the safety and smoothness of the ride. For more information on

how the ACC works, please refer to Section 7.2.3.1. For context on the simulator that was developed for training this algorithm, please refer to Section 7.2.3.2.

The ACC-based controller is an RL controller that is trained using Proximal Policy Optimization [27]. Elements of training the Markov Decision Process (MDP) problem [124] are described below:

Observation Space

- v, velocity of the AV.
- v_s, target speed given by the Speed Planner.
- l, a "minicar" flag that indicates whether there is a leader vehicle detected, nominally within 80 meters (dubbed as such due to the miniature car icon that appears in the dashboard when the leader is detected).
- s, the current ACC speed setting
- g, the current ACC gap setting

Action Space

- The requested ACC speed setting, which our onboard computer will realize in a series of button presses (see Figure 7.6). The speed setting dictates the maximum speed at which the ACC can drive.
- The requested ACC gap setting, which the onboard computer will realize with priority over the speed setting. The gap setting takes on three bars between one and three, with each bar indicating a higher allowable gap. Each bar roughly corresponds to constant time gaps of 1.2, 1.5, and 2.0 seconds.

In addition, a clipping mechanism was used to ensure controller safety and social acceptability, particularly in the absence of leader state information. The *post facto* lower and upper bounds placed on the speed setting output are based on the average speed of the ego vehicle during the last one second (ten timesteps, with 0 mph and null observations omitted):

$$\begin{aligned} v_{\text{lower}} &= \frac{1}{10}\sum_{i=1}^{10} v_i - 15\text{mph} \ , \\ v_{\text{upper}} &= \frac{1}{10}\sum_{i=1}^{10} v_i + 5\text{mph} \ , \end{aligned} \tag{7.20}$$

where v_{lower} and v_{upper} are the lower and upper bounds of the clip, respectively. The final clip is executed as:

$$\begin{aligned} v_{\text{clip}} &= \min(\max(v_{\text{action}}, v_{\text{lower}}), v_{\text{upper}}) \\ v_{\text{final}} &= \min(\max(v_{\text{clip}}, 20\text{mph}), 73\text{mph}) \ , \end{aligned} \tag{7.21}$$

where v_{final} is the final speed setting command and v_{action} is the un-normalized speed output, or action, from the neural net. The 73 mph maximum speed accounts for modest speeds in excess of the posted speed limit of 70 mph.

Energy Models The quantification of the energy demand of the vehicles on the road, given their trajectories, requires vehicle-specific energy models that take as an input the velocity profile $v(t)$ and the road grade profile $\theta(t)$ and output the resulting energy/fuel consumption rate $f(t)$. This project requires the quantification of the energy demand of traffic flow at large, composed of many vehicles; and also the use of reinforcement learning and optimization techniques that minimize (under certain constraints) the energy demand of traffic. Thus, the energy models used should accurately represent different vehicle types on the road and should average out any local non-convexity behavior due to gear switching, to avoid trapping the optimizer in local minima. For that purpose, we use energy models derived from a systematic model-reduction procedure to generate simple fitted models. The procedure starts from the fidelity software Autonomie [202], for a number of vehicles, each of which represents a typical average vehicle of a given class.

To capture the diversity and prevalence of different vehicle types on US roads we select a representative group of vehicle classes on which we apply the model-reduction process to derive their corresponding simplified energy models. Those vehicle classes are divided into two categories: (1) light-duty vehicles: compact size sedan, midsize sedan, midsize SUV, and Pickup, and (2) heavy-duty vehicles: Class4PND (Pickup and Delivery) and Class8Tractor. Each vehicle model represents a class of vehicles that have comparable weight (with load assumed half full) and fuel consumption characteristics [203].

We use the simulation software Autonomie Rev 16SP7 [202] with a library of energy models for several types of vehicles, including a detailed plant and controller model for its components in MATLAB and Simulink, where the blocks and files can be customized. To build our models, we use (i) physics-based vehicle parameters extracted directly from Autonomie, (ii) tuned parameters extracted in an automated fashion by running the Autonomie model on test cycles, and (iii) performance maps computed gear-by-gear on a complete velocity–load phase space of driving by running Autonomie's customized vehicle models on a VCD (those maps are vehicle speed to engine speed, vehicle speed and wheel force to engine torque, and engine speed and torque to fuel rate).

We build an energy model that is semi-principled in that it has a physics-based part using Autonomie's extracted physics-based vehicle parameters, but it also relies on the maps obtained from the VCD. Gear scheduling in this model is based on choosing the feasible gear that yields the minimal fuel consumption, and the torque converter bypass clutch is

assumed open in the first gear only. In contrast to Autonomie that considers hysteresis effects, this model yields the fuel consumption rate (and other outputs) as a direct function of instantaneous velocity v, acceleration a, and road grade θ.

A further model-reduction step is conducted by fitting the semi-principled models into simplified models. Those models have a simple polynomial structure that can easily be integrated into optimization and control problems, yet they are highly accurate. The fuel consumption rate function is

$$f(v,a,\theta) = \max\left\{\beta,\ C(v) + P(v)a + Q(v)a_+^2 + Z(v)\theta\right\}\ , \tag{7.22}$$

where β is the minimum fuel rate set to be zero or a positive constant depending on the fuel cut criteria, $a_+ = \max(-\frac{P(v)}{2Q(v)}, a)$, $C(v) = c_0 + c_1 v + c_2 v^2 + c_3 v^3$, $P(v) = p_0 + p_1 v + p_2 v^2$, $Q(v) = q_0 + q_1 v$, and $Z(v) = z_0 + z_1 v + z_2 v^2$.

In the above functions, c_0 ensures that fuel is being consumed at idle, the c_1, c_2, and c_3 terms can be interpreted as fuel consumed due to friction and air-drag, the $P(v)$ term yields fuel demand due to non-zero accelerations, the $Q(v)$ term captures the super-linear trend of fuel rate with respect to a, and the $Z(v)$ term captures the fuel consumed due to road grade, with the z_1 term playing the role of the weight force exerted at θ. Both types of models are validated, for all different vehicle types, against Autonomie models as the ground truth on standard EPA drive cycles [204] for flat roads and constant road grades drive cycles, and the results showed that the models are highly accurate (within 4% for zero road grades). See [205] for more details on the design of the energy models and corresponding results.

Reward Function The reward function is:

$$\begin{aligned} r_t =& 1 - c_1 a_t^2 - c_2 (v_t^{av} - v_t^{sp})^2 + ... \\ & - \frac{c_3}{n} \sum_{i=1}^{n} E_t^i - c_4 \mathbb{1}_{h_t^{av} \le h_{min} \vee h_t^{av} \ge h_{max}}\ , \end{aligned} \tag{7.23}$$

where $c_1 - c_4$ are coefficients, a_t^2 is an acceleration penalty; $(v_t^{av} - v_t^{sp})^2$ is a squared penalty on the difference between the Speed Planner's suggested speed and the actual speed; $\frac{1}{n}\sum_{i=1}^{n} E_t^i$ is instantaneous fuel consumption according to the energy models described above and the last indicator term is an intervention penalty that is invoked if the space gap is less than the minimum space gap or greater than the maximum space gap.

Further details on the development and design decisions for the state and reward representations and the intermediate and final policies are given in depth in [132].

7.4 Candidate Controller Selection

In the months leading up to the MVT open road test, our team developed a multitude of candidate controllers, as seen in previous sections and sidebars. In order to pare down the

list of candidate controllers to be considered for real-world deployment in November 2022, various (Speed Planner, ACC-based Vehicle Controller)-combinations were assessed over a range of simulation scenarios. The simulation and testing framework follows from [206] and features an updated simulation methodology described in Section 7.2.3.2.

Simulation scenarios included shockwaves, bottlenecks, and freeflow. Shockwave and freeflow scenarios follow precisely from the simulator described in Section 7.2.3.2 and only differ in the leader trajectory (one being stop-and-go, and the other being all high speed). The bottleneck scenario features a dynamically imposed speed limit in a spatial region of the domain. The speed limit is inversely related to the vehicle density of the region, and the scaling parameter is tuned to historically observed speeds in the I-24 region. The bottleneck region is meant to model a weaving area where a close proximity of on-ramps and off-ramps results in increased lane-change frequency.

The key performance indicators (KPIs) used to assess the controllers include fuel economy, throughput, and network speed. Fuel economy is the overall miles per gallon of all vehicles, and it is computed by applying the energy model and computing

$$\text{fuel economy} = \frac{\sum_{t=0}^{T}\sum_{i=1}^{n} E_t^i}{\sum_{i=1}^{n} x^i}\ , \tag{7.24}$$

where the numerator is the total fuel consumed by all vehicles over all simulation time, and the denominator is the total distance traveled by all vehicles. The throughput is measured by counting the number of vehicles crossing various positions along the highway, normalized by time. For the shockwave and freeflow simulations, this is taken to be the straight average of the time-average of five equi-spaced measurement locations. For the bottleneck, this is taken to be the steady-state (or final value) at the measurement location just downstream of the bottleneck region. The network speed KPI is defined as the total distance traveled by all vehicles divided by the total driving time of all vehicles.

At the time of our self-imposed "controller freeze" (that is, the time at which substantial changes to controllers are no longer allowed), there were two Speed Planner variants and 12 Vehicle Controller variants. KPI performance for each of these controllers is shown in Figure 7.7. The Speed Planner variants are kernel smooth (just the first three steps described in Speed Planner) and kernel smooth with RL buffer (all steps described in Speed Planner). The Vehicle Controller variants include:

- **Simple**: a hand-designed logic-based controller that largely adheres to the Speed Planner suggestion.
- **MicroAccel**: the main controller described in Acceleration-Based Controller with modifications to output ACC set points. This controller itself has six variants with different parameter choices.
- **RL**: the RL controller described in ACC-Based Controller. This controller has five variants with different training meta-parameter choices.

In the shockwave scenario, an ideal controller should maximize fuel economy improvements and remain neutral on throughput and network speed. In the bottleneck scenario, an ideal controller should maximize throughput improvements and optionally improve or stay neutral on the other two KPIs. In the freeflow scenario, an ideal controller will not worsen on any of the KPIs. Given these criteria, we determined that the HybridRL Vehicle Controller paired with the Kernel Smooth with RL Buffer Speed Planner presented the ideal combination for the ACC-Based vehicles.

7.5 Open Road Field Operational Test

Following the controller selection, this section introduces the implementation details of the selected controllers, experimental design for the 100 AV deployment, and contemporaneous data collection procedures.

7.5.1 Server-side implementation

As indicated in Figure 7.4, the server-side implementation includes the construction of the database, the API, and the Speed Planner algorithm scripts. The database schema is designed following the star schema data model, consisting of dimension tables (for storing static metadata) and fact tables (for storing quantitative data). The tables include:

- `dim_vehicle` for storing the vehicle metadata, route and lane assignment;
- `dim_inrix_segment` for storing INRIX road segment data for I-24;
- `dim_i24_segment` for storing finer-grained road segment data, which supports TSE and Speed Plan profiles after data fusion;
- `fact_vehicle_ping` for storing the realtime AV ping information;
- `fact_vehicle_observation` for storing extracted vehicle observations from the AV ping table;
- `fact_inrix_estimate` for storing realtime INRIX data;
- `fact_speed_planner` for storing Speed Plan profiles.

The `fact_vehicle_ping` and `fact_inrix_estimate` tables are continually being populated with realtime updates at 1Hz. After processing these input data via the Python-based Speed Planner algorithm, the Speed Plan is written to the `fact_speed_planner` table, which is exposed to the internet via a PHP-based HTTP API.

7.5.2 Vehicle-side implementation

The vehicle-side implementation includes several software libraries and a mix of custom and off the shelf hardware components.

In the past decades , vehicles have been gradually increasing the implementation of the open Controller Area Network (CAN) protocol for in-vehicle networking communication between various electrical modules. The usage of the CAN bus lets automotive manufacturers use minimal wiring to design complex systems involving anything from engine diagnostics, infotainment, security, emissions, and (in more recent years) adaptive cruise control (ACC) and Lane Keep Assist (LKA). The ACC system on cars typically involves a sensor module to measure the leading vehicle dynamics and a separate controller module, which communicates with the vehicle's transmission and engine. Since these modules are physically located in different parts of the vehicle they communicate using CAN busses, where listening CAN analyzers can record the information. Figure 7.8 shows the method of tapping into a CAN bus for message reading.

While automotive manufacturers provide information on the structure of the in-vehicle network in the form of wiring diagrams, specific information being sent on the CAN bus is typically a closely guarded secret, likely due to both trade secrets and safety concerns. However, some companies (such as comma.ai [207] and Intrepid Control Systems [208]) are entering the space of custom vehicle autonomy by selling modules that intercept CAN messages between these modules and provide their own inputs based on custom controllers. This is only possible after spending time decoding messages and reverse engineering the protocols needed to inject custom messages. Compared to basic CAN reading, Figure 7.8 shows the different electrical architecture needed for CAN message interception and injection between OEM modules.

In this work our solution is based on a similar principle, however additional protocols on the 2023 Nissan Rogue (which comprise the vast majority of our control fleet) prevented the injection of commands, though CAN data could still be read for realtime controller inputs. Due primarily to supply chain issues and additionally to an inability to inject CAN messages, a custom circuit board was developed, coined mattHAT (Hardware Attached on Top). A mattHat can be seen in Figure 7.9. The mattHAT acted as a CAN interface and also provided methods of sending ACC button press commands using custom circuitry. The board was designed to plug into a Raspberry Pi 4. Custom wire harness cables were designed and built for vehicle installation across the 97 Nissan Rogue vehicles, targeting specific CAN busses related to the vehicle's ACC state.

A C++ based library named `libpanda` [122] was developed to operate the mattHAT's CAN interface and button spoofer as well as record raw CAN and GPS data. `libpanda` also supports a USB-based GPS module to both record the vehicle's position and to synchronize the system's time. A tool named Strym was also developed to decode and quickly analyze libpanda's recorded data for CAN signal decoding and classification [209, 210, 211]. To allow control designers to easily make use of `libpanda`, the Robot Operating System (ROS) was installed on the Raspberry Pi. A set of ROS nodes were designed in the software project

named `can_to_ros` [125, 191], which abstracts various sensors and actuators into a set of ROS topics. A tool named `bagpy` was also developed to quickly process and plot recorded ROS data, in the format of bagfiles [212]. With the infrastructure of ROS in place, control designers could use modeling software like MATLAB's Simulink to design controllers and generate code to greatly ease controller integration.

`libpanda` also features a set of auxiliary services to manage the vehicle at scale. A method to automatically perform over the air (OTA) updates was implemented so that vehicles would not need to be manually handled. An additional server integration named `piStatus` was built so that the system could regularly post its status to easily assess the system's health. `piStatus` would be monitored regularly to ensure vehicles pass status checks before deployment. `piStatus` would be continuously monitored throughout each experiment to note any hardware issues so that vehicles could undergo maintenance before the next deployment. A shutdown script also worked in tandem with an off-the-shelf battery-backup Pi HAT to perform automatic data uploads at the end of an experimental drive when the vehicle was shut off. Figure 7.10 shows a WiFi antenna installed in the center of the vehicle parking lot for granting each vehicle's embedded computer internet access for data upload and OTA updates. Without these set of services, managing and maintaining a project at this scale would have been impossible.

7.5.3 I-24 MOTION

The CIRCLES team deployed the vehicle controllers on I-24 southeast of Nashville, TN due to the recent creation of a new testbed, known as I-24 Mobility Technology Interstate Observation Network (MOTION) [180, 181]. I-24 MOTION is a four-mile section of I-24 designed to produce ultra-high resolution trajectory data of all vehicles on the roadway for the purposes of traffic science and experimentation on automated vehicles and traffic management. The system produces trajectory data (Figure 7.11) using 276 cameras (Figure 7.12) on fixed roadside poles between 110–135ft tall, to minimize visual occlusion. The debut of the I-24 MOTION testbed coincided with the live CIRCLES experiment, for which the testbed is uniquely suited to gather data on the traffic stream impacts of a fleet of test vehicles.

Processing of raw video into vehicle trajectories by I-24 MOTION happens in two stages. 1) A computer vision pipeline [214, 181] makes the initial vehicle detection and type classification from the raw video [214], including vehicle dimensions by using 3-D bounding boxes. Vehicle classes used by the object detection are: sedan, midsize, pickup, van, semi, truck, motorcycle. It tracks detected vehicles across adjacent camera views in the vehicle's direction of travel. The computer vision processing is distributed across ten servers with contiguous groups of cameras allocated to each. Vehicles are not tracked between servers, so a vehicle trajectory covering the length of the testbed consists of at least ten fragments; additional fragmentation may occur due to unavoidable occlusion (for example, by overpasses). The computer vision pipeline also converts image space coordinates (where vehicle detections occur) into a roadway coordinate system using a homography transformation calibrated every hour to each camera. 2) Post-processing algorithms [215] first stitch fragmented trajectories together using an

online minimum cost network flow graph problem. Each trajectory is then subject to a reconciliation procedure to ensure feasible and smooth higher order dynamics (acceleration and jerk), formulated as a quadratic program.

Processing of raw video into vehicle trajectories by I-24 MOTION happens in two stages. 1) A computer vision pipeline [214, 216, 217] makes the initial vehicle detection and type classification from the raw video, including vehicle dimensions by using 3-D bounding boxes. Vehicle classes used by the object detection are: sedan, midsize, pickup, van, semi, truck, motorcycle. It tracks detected vehicles across adjacent camera views in the vehicle's direction of travel. The computer vision processing is distributed across ten servers with contiguous groups of cameras allocated to each. Vehicles are not tracked between servers, so a vehicle trajectory covering the length of the testbed consists of at least ten fragments; additional fragmentation may occur due to unavoidable occlusion (for example, by overpasses). The computer vision pipeline also converts image space coordinates (where vehicle detections occur) into a roadway coordinate system using a homography transformation calibrated every hour to each camera. 2) Post-processing algorithms first stitch fragmented trajectories together using an online minimum cost network flow graph problem [218]. Each trajectory is then subject to a reconciliation procedure [215] to ensure feasible and smooth higher order dynamics (acceleration and jerk), formulated as a quadratic program. A data visualization library [213] assists with interpreting the datasets generated by the system.

Due to the critical nature of data from the MVT, the I-24 MOTION system retained a secure backup of the experiment data and baseline traffic conditions. Raw I-24 MOTION imagery is not accessible outside of the I-24 MOTION administrators, per the testbed's privacy policy. However, this backup allows re-processing of the video and/or the raw vehicle detections from the computer vision pipeline as data generation pipeline advances. The current released version of the data has gone through re-processing rounds to address known errors and limitations, including using hourly re-calibrated homography transformations for camera perspective [217].

7.5.4 Experimental Design

The live traffic experiment took place on I-24 in November 2022. With 100 vehicles to deploy, over 150 drivers were hired from local colleges, security guards, delivery drivers, team members' relatives, and elsewhere. The drivers were trained to adhere to specific assigned routes and lanes and to activate our custom ACC system whenever driving in their assigned lanes. Safety features were implemented such that the controller would default to stock ACC behavior if engaged off of the highway.

The specific routes driven by the AVs were chosen to maximize the penetration rate where the I-24 MOTION system [180] would capture the effect on the surrounding commuters (bulk traffic). Initially, a single-loop route that would have the AVs circulating from 7:30am to 9:30am was considered. It was determined that this would likely cause increased congestion on the Exit 57 off-ramp, leading to extended queuing and potential spillback onto I-24. For this

reason, we divided our AVs into two groups with the two different routes partially overlapped on I-24.

The AVs are released at 6:00am in order for all 100 to be deployed by the target time 7:30am. The drivers are instructed to repeat their driving routes for 2 hours, or sooner if they wanted a break. On a return trip, they exit the highway at Exit 60 (Hickory Hollow), to return to the Field Headquarters parking lot and return their car keys at a desk before leaving the lot. As AVs return to the lot for a break, other drivers (already on break) head down to a queue to come out to the lot when needed. With this smoothly running rotation system, we were able to keep the 100 AVs on the routes during peak hours, while keeping track of their locations in real time. For more details about the routes, and logistical choices made for the experiment, driver training, the daily schedule, or the penetration rate estimates, see [219].

7.5.5 Data Release

As a part of this publication, we are releasing to the public a dataset from the MVT. This data release will include data collected by the AVs as well as the I-24 MOTION [181] system. See also [214, 217, 216] for other datasets generated by I-24 MOTION.

The data contains GPS and CAN data collected by the 100 AVs from all testing days. The longitudinal position along the highway is modified *post hoc* with an offset to account for systematic positioning error (for example, caused by placement of the GPS unit within the vehicle). The offset is determined by finding all likely matches in the I-24 MOTION dataset, and computing the median delta. All other data is provided unaltered except for interpolating onto a resampled, fixed 10Hz time grid. Lane identification is not provided; rather, the driver's assigned lane can reasonably be assumed to be the true lane when the controller is engaged.

The raw I-24 MOTION trajectory information is processed [215, 218] and then is appended with additional information. Speed and acceleration are inferred from the longitudinal position (which, as described previously, is processed to be sufficiently smooth for second-order derivatives). Road grade information is added via a parametric road grade map model. Fuel information, including various fuel rates and those for reference trajectories, are appended through the application of fuel rate models (see [205]). Fuel rate models are selected with a one-to-one mapping of vehicle classes to fuel model classes. For each I-24 MOTION trajectory segment, a reference trajectory is constructed that matched the initial and terminal positions and speeds and minimize the functional $\max_t a(t) - \min_t a(t)$ subject to the additional constraint $v(t) \geq 0$. This functional is a simple vehicle-independent proxy for the more complicated energy models constructed in [205], motivated by the following arguments: (a) due to the super-linear growth of fuel rate with increasing acceleration, peak accelerations should be minimized; and (b) unnecessarily heavy braking results in wasted fuel, hence peak braking (which is $-\min_t a(t)$ for typical trajectories) should also be minimized. By Pontryagin's maximum principle, this generally results in piecewise-constant acceleration profiles with a jump at the trajectory's midpoint in time. Lastly, using the implicit lane and

longitudinal matching with the AV dataset, we also supply relative distances and vehicle IDs for nearest upstream/downstream, engaged-or-not AVs to assist in subsequent analyses.

Further information on the structure of the data will be provided in the data documentation released with the data.

7.6 Results

The experimental, observational, data collection, and data processing framework described above generated a large amount of data, capturing every single vehicle on a highway during the deployment of 100 controlled vehicles. Analogous to previous seminal traffic data sets, such as NGSIM [220], the new data are expected to inspire and enable many subsequent findings. To highlight this potential, we here present some key first findings and insights, based on an analysis of the data with a macroscopic perspective.

We construct macroscopically meaningful fields in time–space, most prominently a field that shows the energy (in)efficiency of traffic at large on the I-24 highway segment. This is achieved by applying Edie's method [221] on boxes of size $h_t \times h_x$, where $h_t = 10$s and $h_x = 200$m, to the I-24 MOTION trajectories to construct the following fields:

- vehicle density $\rho(t, x)$, as the total vehicle time spent in each box, divided by the size of the box, $h_t \cdot h_x$;
- flow rate $q(t, x)$, as the total distance traveled in each box, divided by $h_t \cdot h_x$; and
- fuel rate density $f(t, x)$, as the total fuel consumed in each box, divided by $h_t \cdot h_x$.

From these fields, other meaningful fields are obtained, such as the bulk velocity field $u(t, x) = q(t, x)/\rho(t, x)$, the bulk fuel rate $\phi(t, x) = f(t, x)/\rho(t, x)$, and the bulk fuel consumption $\psi(t, x) = f(t, x)/q(t, x)$. The latter quantity $\psi(t, x)$, measurable for instance in grams per meter, represents the fuel demand per distance traveled of all vehicles in the $h_t \times h_x$ vicinity of the position (t, x). Figure 7.13 shows $\psi(t, x)$ for two experimental days: Wednesday November 16 and Thursday November 17, 2022. Each plot is overlaid with the trajectories of all control vehicles. This represents the first time that a complete time–space diagram of the energy inefficiency of traffic, based on accurate trajectories of *all* vehicles on the roadway, has been provided.

In the same spirit as the purely microscopic Figure 7.11, one can, for both days shown in Figure 7.13, clearly see the traffic waves traveling backwards along the highway, as well as the increased fuel consumption incurred in these waves. The figure also shows the increased fuel demand in the uphill segment between $x = 5$km and $x = 6$km, and the reduced fuel demand in the downhill segment thereafter. The AV trajectories are colored red when the automated controller was activated, and white when the vehicle was under human control. The two shown test days were quite different in terms of the engagement rates of the controllers: on 11/16, the controllers were engaged 38% of all times, while on 11/17, the engagement rate went up to 78%. This difference was caused by a combination of increased driver comfort

with the automation and a more reliable communication of traffic information to the vehicles on 11/17.

Given the low penetration rate of the control vehicles on the highway, it was not expected that they would completely smooth out all traffic waves—and the plots in Figure 7.13 confirm that expectation. However, the AVs may still have had some positive contribution on the energy efficiency of the flow at large. Whether that was in fact the case, we first note that the macroscopic plots like in Figure 7.13 allows for a targeted inspection of different regions of interest in time-space. For instance, on 11/16, there is a distinct region of high fuel inefficiency, around time 08:25:00 and location 5.0km–6.5km; and notably, this high-fuel region coincides with all AV controller being inactive. In contrast, on 11/17 such clusters of inactive AVs did not occur—and the fuel consumption map does not exhibit similarly large high-fuel regions. Another notable anecdote occurs on 11/16 in the wave that goes through $x = 2.3$km at time 07:00:00. First two active AVs notably dampen the wave; then the wave keeps on growing while four inactive AVs run through it (at $x = 3$km); followed by several active AVs (around $x = 4$km) notably dampening the wave again.

Many more quantitative results and plots are available in the publications of the different teams that contributed and deployed a controller during the MegaVanderTest [201, 167, 132]. Since each team deployed their controllers in different manners, on different vehicles, or at different scales, the results are not easily comparable—which leads to interesting analyses from different points of views. Each team has detailed the process they went through to clean, parse, and extract relevant information from the data, in order to estimate their controller's impact in terms of energy savings and traffic smoothing. Continued effort to quantify our vehicles' overall impact is underway and planned for future publication(s).

7.7 Conclusion

This work describes the control architectures and implementations of a 100 automated vehicle deployment to improve traffic efficiency on a freeway using a small fraction of automated vehicles. It is the largest field experiment to use CAVs to regulate the overall traffic flow, and the deployment strategy enabled algorithms from diverse fields spanning model based control to reinforcement learning. The control strategy presented in this work was a hierarchical control approach in which the upper level speed planner provided target velocities to a lower level control law responsible for performance and safety. These algorithms were deployed in the largest field experiment of its kind, on a heterogeneous vehicle fleet using low cost vehicular instrumentation. The data from the vehicles were combined with datasets generated from I-24 MOTION [180, 181], providing a large data resource for further study on the interaction of control vehicles on bulk traffic flow.

While the MegaController experiment demonstrates the feasibility and benefits of large-scale autonomous vehicle traffic control, it also highlights opportunities for improving the underlying computational methods that enable such deployments. In particular, the speed planning component relies on relatively coarse traffic state estimation and prediction methods

that could benefit from more sophisticated approaches to modeling traffic flow dynamics. The integration of physics-informed learning methods with conservation-law traffic models presents a promising direction for enhancing the predictive capabilities that could further improve the performance of future large-scale deployments, which motivates the methodological development presented in the following chapter.

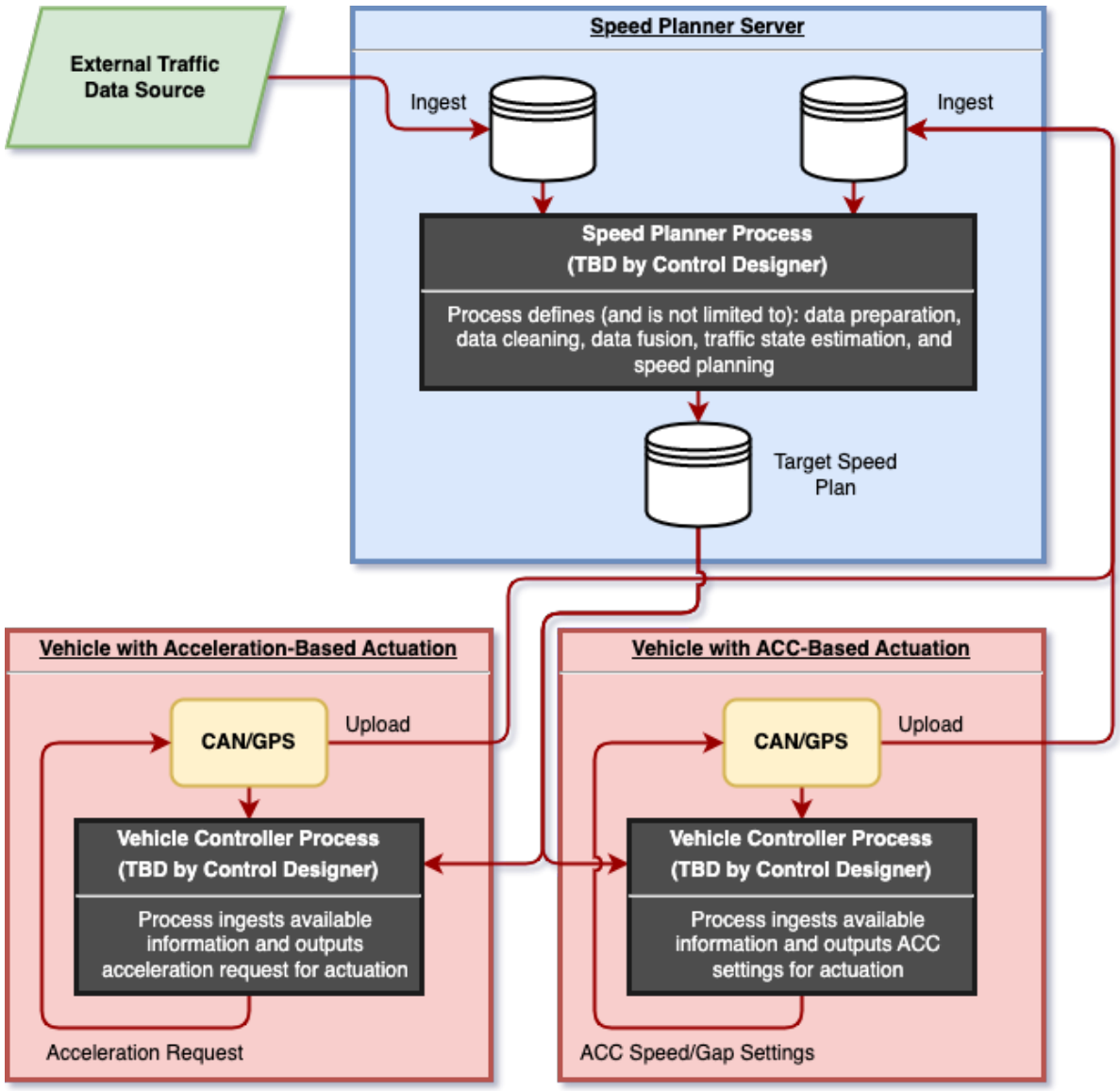


Figure 7.1: Architectural framework of the MegaController. The hierarchical and modular nature of the design allows for greater flexibility in design decisions and dealing with varied sensing and actuation capabilities of the heterogeneous fleet. The blue box represents the centralized Speed Planner unit, and the red boxes represent decentralized Vehicle Controllers, which are vehicle-dependent (that is, each vehicle has a different control architecture and thus requires a different control paradigm). The components work in concert to achieve higher level goals of flow smoothing.

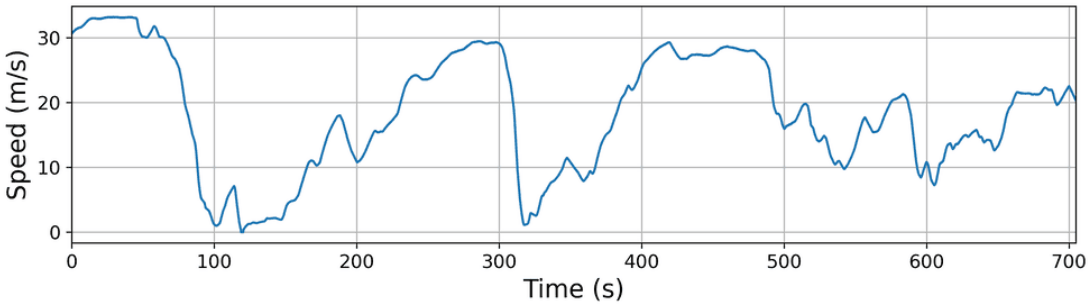


Figure 7.2: Speed vs. time for one of the dataset trajectories, exhibiting large acceleration and breaking patterns that can typically lead to stop-and-go waves.

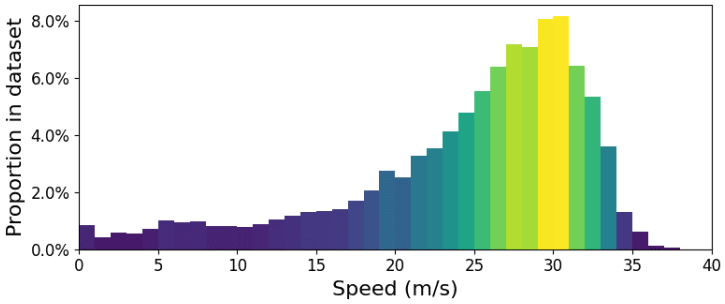


Figure 7.3: Distribution of speeds in the I-24 dataset.

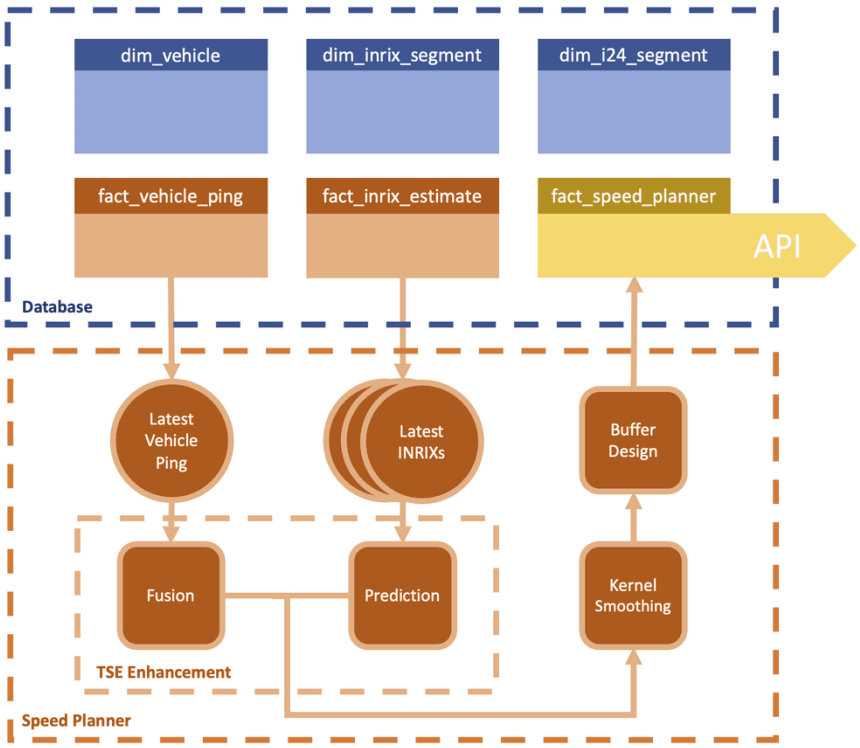


Figure 7.4: Data pipeline and major function modules of Speed Planner. At the beginning of each update, the Speed Planner extracts a combination of macroscopic TSE and vehicle observations from the corresponding factual tables (`fact_inrix_estimate`, `fact_vehicle_ping`) in the database to calculate the target speed profile. The raw TSE is used as the input of the prediction module, of which the output is fused with vehicle observations. The fusion is then smoothed and used in the buffer design module, of which the output is saved into the database (`fact_speed_planner`) and published as the target speed profile.

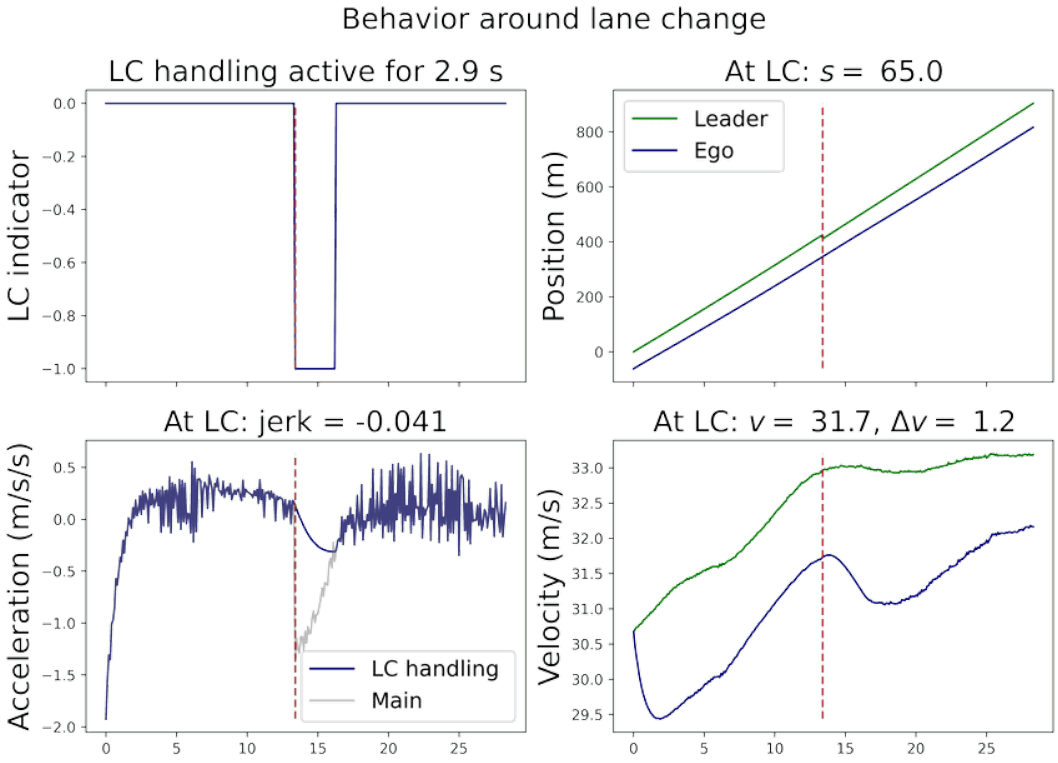


Figure 7.5: An example of the effect of the lane-change handling mechanism on a real-world trajectory in the event of a cut-in. *Top left*: lane-change event is detected and considered safe so the lane-change recovery controller was active for 2.9s. *Top right*: a car cuts-in in front of the ego vehicle at a headway of 65m. *Bottom left:* the main controller commanded acceleration drops sharply due to the lane-change event causing a large jerk value, but the lane-change controller smooths out this drop in the acceleration. Bottom right: The relative velocity is large enough at the lane-change allowing the controller to be active.



Figure 7.6: A photograph of a Nissan Rogue's steering wheel buttons that control the vehicle's ACC. The ACC system is turned on by manually pressing the blue icon on the far right. Through our vehicle interfacing efforts, we are able to electronically press the + and − buttons to toggle the ACC speed setting up 1 mph and down 1 mph, respectively, or hold them to increment by 5 mph. The three ACC gap settings are rotated through by pressing the button on the bottom right.

SHOCKWAVE SCENARIO		Speed Planner					
		Kernel Smooth			Kernel Smooth with RL Buffer		
		Fuel Economy % Increase	Throughput % Increase	Speed % Increase	Fuel Economy % Increase	Throughput % Increase	Speed % Increase
Vehicle Controller	Simple	17.33%	1.18%	-0.32%	24.24%	-0.04%	-5.86%
	MicroAccel-a	9.33%	-0.08%	-0.37%	10.71%	1.18%	-0.75%
	MicroAccel-b	4.56%	0.26%	7.68%	8.87%	2.27%	2.87%
	MicroAccel-c	9.33%	-0.08%	-0.37%	10.71%	1.18%	-0.75%
	MicroAccel-d	9.33%	-0.08%	-0.37%	10.71%	1.18%	-0.75%
	MicroAccel-e	9.33%	-0.08%	-0.37%	10.71%	1.18%	-0.75%
	MicroAccel-f	12.20%	1.02%	0.37%	14.94%	0.35%	0.45%
	RL-a	14.31%	-1.11%	-1.72%	13.25%	-1.32%	-1.69%
	RL-b	12.38%	-1.10%	-1.87%	11.63%	-1.45%	-1.87%
	RL-c	-2.76%	0.82%	0.37%	-1.42%	0.36%	0.15%
	RL-d	3.24%	0.26%	0.67%	-0.53%	0.11%	-0.27%
	RL-e	23.18%	2.84%	-2.19%	18.47%	2.04%	-1.69%
	HybridRL	22.96%	2.75%	-2.29%	17.49%	1.69%	-2.14%

BOTTLENECK SCENARIO		Speed Planner					
		Kernel Smooth			Kernel Smooth with RL Buffer		
		Fuel Economy % Increase	Throughput % Increase	Speed % Increase	Fuel Economy % Increase	Throughput % Increase	Speed % Increase
Vehicle Controller	Simple	6.85%	13.57%	8.76%	8.44%	14.34%	12.63%
	MicroAccel-a	4.90%	12.77%	4.21%	3.97%	13.16%	3.87%
	MicroAccel-b	1.81%	13.55%	8.16%	-0.60%	13.16%	7.99%
	MicroAccel-c	4.90%	12.77%	4.21%	3.97%	13.16%	3.87%
	MicroAccel-d	4.90%	12.77%	4.21%	3.97%	13.16%	3.87%
	MicroAccel-e	4.90%	12.77%	4.21%	3.97%	13.16%	3.87%
	MicroAccel-f	1.26%	13.16%	3.35%	1.09%	12.39%	3.18%
	RL-a	4.25%	9.30%	4.47%	3.45%	9.30%	4.64%
	RL-b	3.83%	9.30%	4.55%	3.36%	9.30%	4.64%
	RL-c	0.88%	13.16%	3.18%	0.88%	13.16%	3.18%
	RL-d	0.91%	12.77%	3.78%	1.06%	8.82%	4.81%
	RL-e	8.22%	14.34%	8.08%	6.72%	14.73%	8.68%
	HybridRL	2.95%	14.73%	9.19%	2.92%	14.74%	9.19%

FREEFLOW SCENARIO		Speed Planner					
		Kernel Smooth			Kernel Smooth with RL Buffer		
		Fuel Economy % Increase	Throughput % Increase	Speed % Increase	Fuel Economy % Increase	Throughput % Increase	Speed % Increase
Vehicle Controller	Simple	1.13%	-0.56%	-2.14%	-2.79%	-1.24%	-1.81%
	MicroAccel-a	14.50%	5.13%	-11.63%	14.52%	5.21%	-11.53%
	MicroAccel-b	13.78%	4.49%	-12.73%	13.80%	5.90%	-11.63%
	MicroAccel-c	14.50%	5.13%	-11.63%	14.52%	5.21%	-11.53%
	MicroAccel-d	14.50%	5.13%	-11.63%	14.52%	5.21%	-11.53%
	MicroAccel-e	14.50%	5.13%	-11.63%	14.52%	5.21%	-11.53%
	MicroAccel-f	15.68%	0.33%	-15.38%	15.81%	0.63%	-15.20%
	RL-a	27.27%	2.50%	-25.19%	25.39%	2.65%	-22.38%
	RL-b	32.76%	2.33%	-23.78%	25.27%	3.47%	-21.54%
	RL-c	-0.19%	1.49%	0.43%	-0.03%	1.60%	0.26%
	RL-d	0.05%	0.12%	-0.86%	0.05%	0.12%	-0.86%
	RL-e	29.47%	1.41%	-34.28%	25.13%	2.35%	-27.52%
	HybridRL	-0.19%	1.49%	0.43%	-0.03%	1.60%	0.26%

Figure 7.7: Summary of controller assessments prior to MVT deployment. Candidate controller components to the MegaController were simulated under three scenarios and evaluated against key performance indicators relative to a baseline simulation of homogeneous human drivers, across various Speed Planners and Vehicle Controllers.

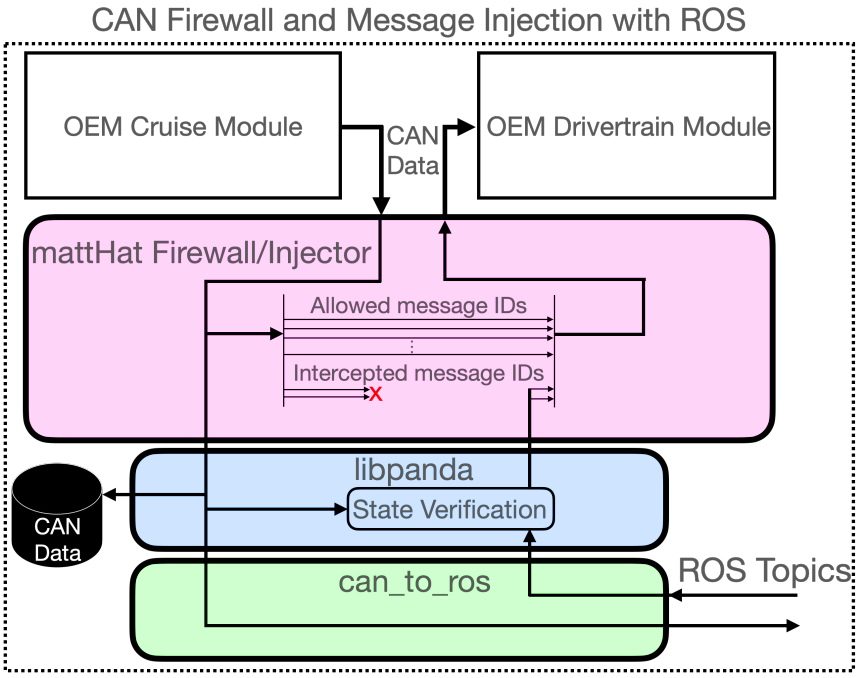


Figure 7.8: CAN firewall and message injection with ROS. A special CAN interface called the mattHat firewalls messages between OEM modules at the hardware level, and can replace blocked messages with third party messages from software with `libpanda`. `libpanda` provides direct CAN recording as well as verification of CAN data to be sent by checking OEM's CAN states. CAN messages can be read and sent through ROS topics provided by adapters in `can_to_ros`.

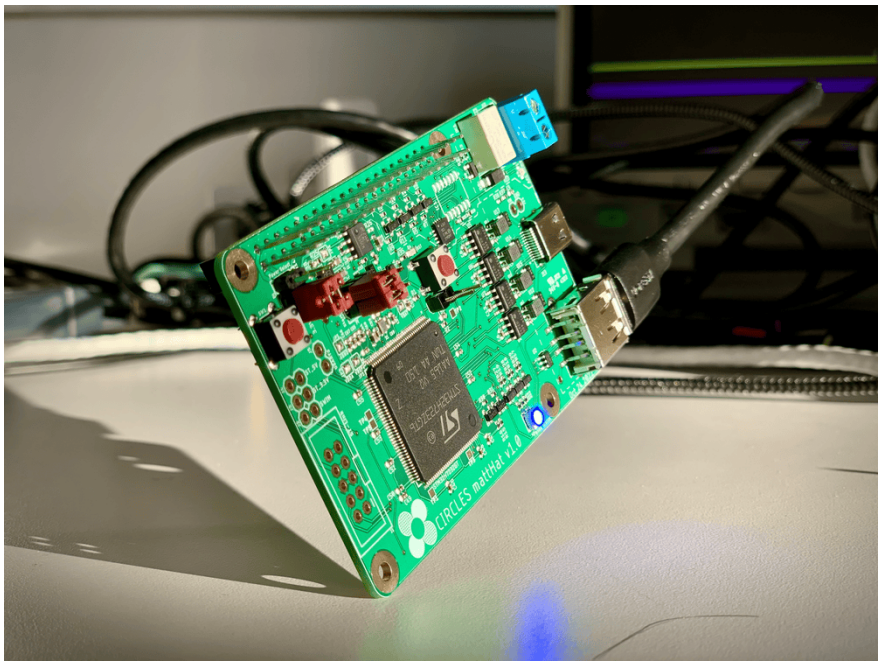


Figure 7.9: The mattHat. This custom designed circuit board was rapidly manufactured for both CAN interfacing and ACC button command spoofing.



Figure 7.10: Parking lot WiFi antenna. A WiFi antenna is placed in the center of the vehicle parking lot, enabling OTA data uploads and software updates for each Raspberry Pi in each vehicle.

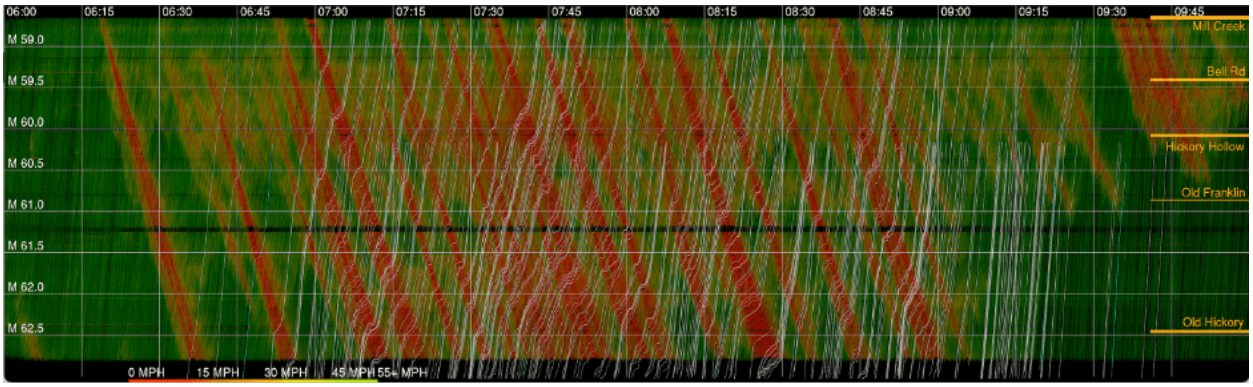


Figure 7.11: Time (horizontal axis) space (vertical axis) diagram generated by I-24 MOTION [180] and associated visualization library [213] during the MVT. Vehicle trajectories driving westbound (up) are colored based on the speed of vehicles (green: freeflow to red: congested). Experiment vehicle trajectories are overlaid in white.

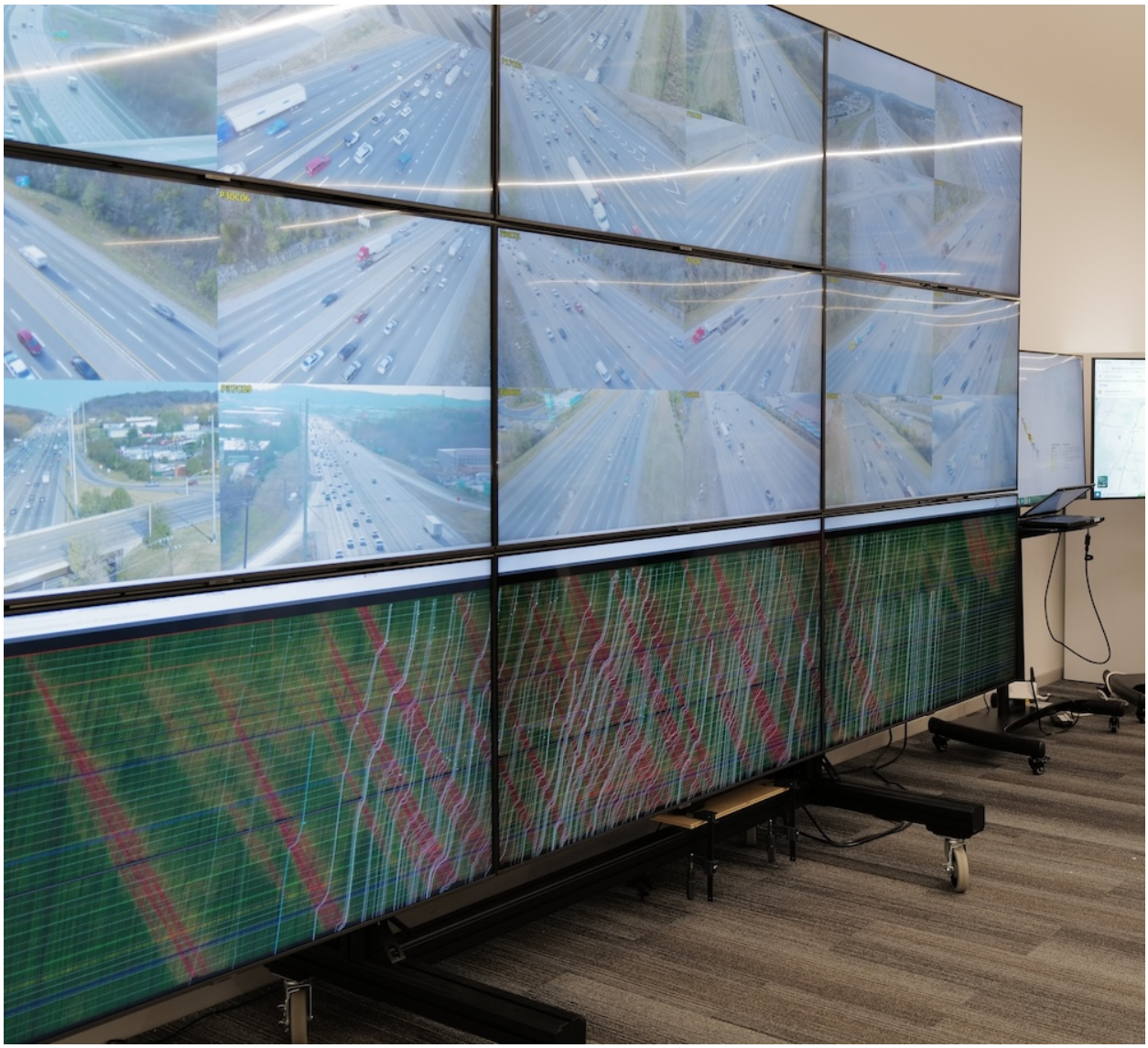


Figure 7.12: Visualization wall in the operation center. The display shows live video feeds from the I-24 MOTION cameras installed along the highway, as well as a time-space diagram generated from the I-24 MOTION and AV data (see Figure 7.11 for more details).

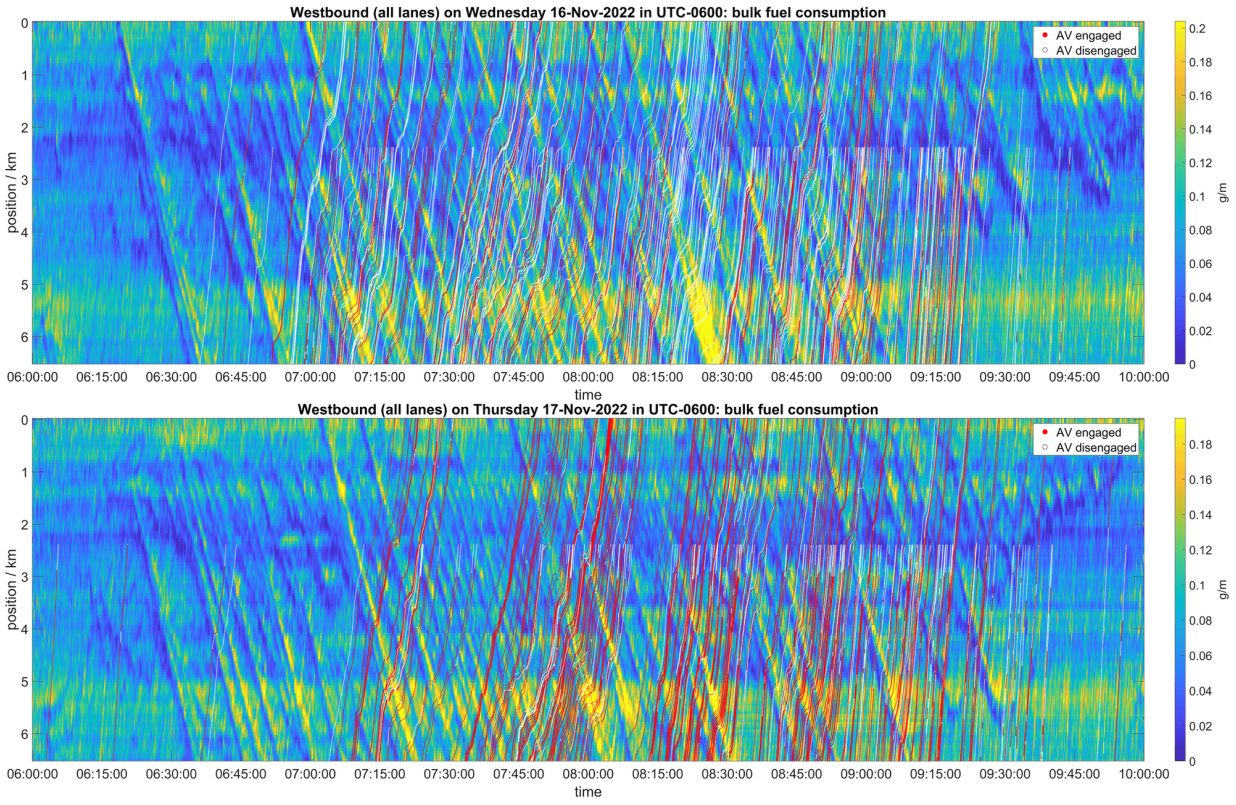


Figure 7.13: Bulk fuel consumption heatmap in time (horizontal axis) and space (vertical axis), based on I-24 MOTION [180] of all vehicles (aggregated over all lanes) driving in the Westbound direction with AV trajectories overlaid (white: controller not engaged; red: controller engaged). Shown are November 16, 2022 (top) and November 17, 2022 (bottom).

Chapter 8

Neural Finite Volume Methods for Hyperbolic PDEs

The large-scale field experiments presented in the previous chapter successfully demonstrated that autonomous vehicle controllers can create measurable improvements in highway traffic flow. However, these deployments also revealed limitations in the traffic prediction and state estimation methods used for speed planning. The MegaController relied on relatively coarse traffic data with significant latency, constraining its ability to anticipate and proactively respond to traffic wave formation. This chapter addresses these limitations by developing enhanced traffic prediction methods that combine the physics-based conservation laws from Chapter 2 with neural network learning capabilities. By treating traffic flow as governed by hyperbolic partial differential equations, we can leverage the mathematical structure of wave propagation while using machine learning to capture complex nonlinear dynamics that classical numerical methods struggle to model accurately. The Neural Finite Volume methods developed here provide a foundation for improving the predictive capabilities of future traffic control systems, representing a natural synthesis of the physics-based understanding and data-driven learning approaches that have guided this research program.

This chapter is based on [222, 223]: Nathan Lichtlé*, Alexi Canesse*, Zhe Fu*, Hossein Nick Zinat Matin*, Maria Laura Delle Monache, and Alexandre M. Bayen. "(U)NFV: Supervised and Unsupervised Neural Finite Volume Methods for Solving Hyperbolic PDEs.", in review, 2025.

8.1 Introduction

The large-scale autonomous vehicle deployment demonstrated in Chapter 7 revealed the potential for coordinated traffic control at unprecedented scale, but also highlighted limitations in the traffic prediction methods used for speed planning. The MegaController relied on aggregate traffic data with substantial latency, constraining its ability to anticipate wave

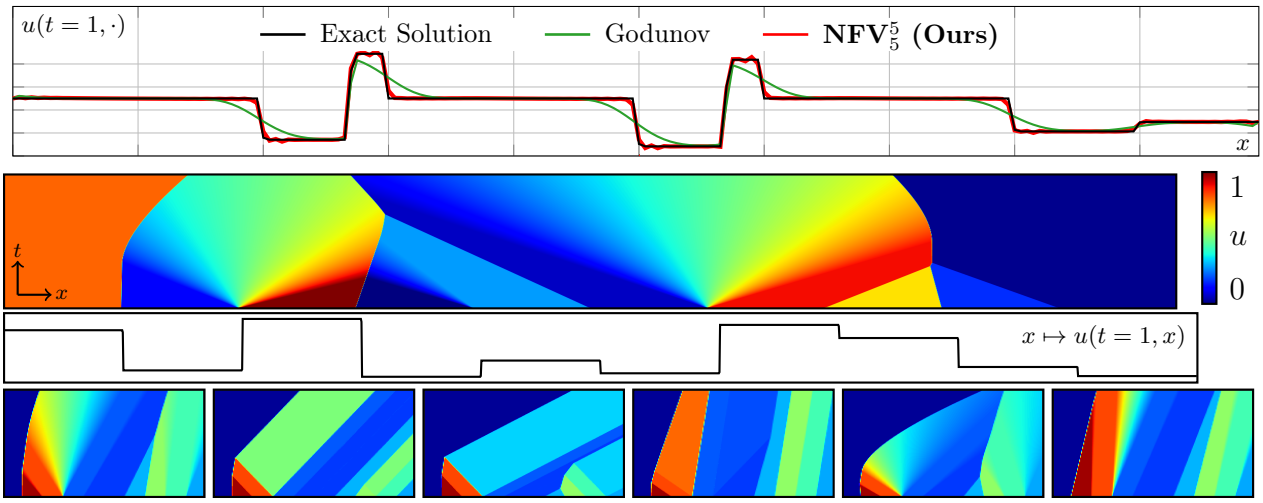


Figure 8.1: **Prediction of entropy solutions of hyperbolic PDEs. Top:** NFV$_5^5$ prediction vs. the Godunov scheme for Burgers' equation at a fixed time. **Mid:** Entropic solution $u(t,x)$ for Burgers' equation over domain $(t,x) \in [0,1]^2$, and its corresponding initial condition. **Bottom:** Entropic solution $u(t,x)$ for the LWR equation with different fluxes sharing the same initial condition.

formation and optimize proactive control strategies. This chapter addresses these limitations by developing enhanced traffic prediction methods that combine physics-based conservation laws with neural network learning capabilities.

As introduced in Chapter 2, traffic flow can be modeled using hyperbolic partial differential equations, specifically the Lighthill-Whitham-Richards (LWR) model that treats vehicle density as a conserved quantity. However, classical numerical methods for solving these equations often lack the accuracy and flexibility needed to capture the complex nonlinear dynamics observed in field data. This motivates the development of Neural Finite Volume methods that preserve the mathematical structure of conservation laws while leveraging machine learning to improve predictive accuracy.

Hyperbolic partial differential equations (PDEs) [224] are fundamental tools to model propagation and transport phenomena with nonlinear or discontinuous behavior, appearing in areas like fluid dynamics and traffic flow. In the present work, we focus on an essential subclass: *conservation laws* [3], which encode the principle that certain physical quantities, such as mass, momentum, or energy, must be preserved over time. A general one-dimensional scalar conservation law takes the form:

$$\partial_t u(x,t) + \partial_x f(u(x,t)) = 0, \tag{8.1}$$

where u is the conserved quantity and f is the *flux function*.

The solutions of hyperbolic PDEs are difficult to approximate due to discontinuities such as shocks (see Figure 8.1), even when starting from smooth initial conditions. Consequently,

classical (smooth) solutions typically cease to exist after finite time, and one must instead rely on weak solutions, Closed-form solutions exist only in rare cases, such as on simple Riemann problems [3] or through the Lax-Hopf formula [225, 226, 227]) in specific concave or convex settings. As a result, practical applications almost always rely on numerical methods for approximating the PDE solution, with finite volume (FV) methods [3] being a popular choice due to their ability to track conserved quantities across discontinuities and capture shock dynamics relatively accurately.

Classical FV methods involve important trade-offs between accuracy near discontinuities, computational cost, stencil size, and implementation complexity. In recent years, neural networks have been explored as flexible and powerful alternatives solvers, showing promise in learning complex dynamics from data or residuals. Yet, many such methods are designed for non-specific models, often at the expense of losing physical structure, including conservation laws and entropy behavior.

We introduce the *Neural Finite Volume* (NFV) method, a modular architecture tailored to conservation laws, that blends the structure-preserving benefits of FV schemes with the expressiveness of neural networks. Conservation is built into the NFV model, using extended spatial and temporal stencils. We develop both a supervised version, trained on solution data from simple cases, and an unsupervised variant (UNFV), which learns directly from the PDE via a weak-form residual loss. This flexibility allows (U)NFV to adapt to the availability of data, leveraging accurate synthetic or field data when present, or solving directly from the equation when solutions are inaccurate or expensive to obtain.

Contributions. Our main contributions are as follows:

- We propose (U)NFV, a neural architecture that generalizes the structure of finite volume methods and thus preserves conservation properties by construction.
- We introduce two variants: a supervised learning one (NFV) and an unsupervised learning one (UNFV), depending on data availability, using either solution data or a weak-form residual loss.
- We demonstrate strong numerical results on several conservation laws, achieving up to 10x lower error than classical FV solvers. Additionally, (U)NFV matches the accuracy of discontinuous Galerkin methods, without their mathematical complexity.
- We show that NFV can be trained on field data that does not necessarily strictly satisfy the conservation law, and still predicts accurate solutions with more flexibility and generalizability than classical solvers.

The remainder of the chapter is organized as follows: Section 8.2 provides a detailed overview of the related work, Section 8.3 recalls the finite volume method and introduces necessary notation, Section 8.4 describes the proposed (U)NFV method in detail, Section 8.5 presents the experiments and results on hyperbolic PDEs, Section 8.6 extends the NFV to experimental field highway data, and Section 8.7 concludes the chapter with discussion on broader impacts. Then, Appendix C.1 provides details about finite volume schemes,

Appendix C.2 illustrates six PDE variants considered in this chapter, Appendix C.3 introduces more general PDE models, Appendix C.4 expands on the experimental data handling and results from Section 8.6, and Appendix C.5 details the experimental setup.

8.2 Related work

Numerical methods. Classical numerical methods for hyperbolic PDEs, such as FV and *discontinuous Galerkin* (DG) [228] methods, are widely used due to their capabilities in capturing shocks and discontinuities. First-order schemes such as the *Lax-Friedrichs* [229] and *Godunov* [230] methods provide robustness but suffer from excessive numerical diffusion, leading to smeared solutions. To address this, higher-order methods like *Essentially Non-Oscillatory* (ENO) [231], *Weighted ENO* (WENO) [232], and DG have been introduced, offering improved accuracy in smooth regions while preserving stability near shocks. DG further improves accuracy through local polynomial approximations but incurs high computational costs [233]. Despite their accuracy, these methods often require extensive manual effort and careful stabilization, motivating the development of flexible, data-driven alternatives.

NN approaches for PDEs. Deep learning has emerged as a powerful alternative for approximating PDE solutions. In supervised settings, neural operators such as *Fourier Neural Operator* (FNO) [234] and *Deep Operator Networks* (DeepONet) [235] efficiently approximate solution mappings from parametric inputs, without requiring explicit mesh discretization in the case of FNO. While successful for general PDEs, these operators have mainly been validated on elliptic or parabolic PDEs, typically characterized by smooth solutions. Conventional neural architectures, such as CNNs [236] for structured domains and GNNs [237] for irregular geometries, have also been adopted as supervised surrogates. However, supervised models rely heavily on large, high-quality labeled datasets, and often lack intrinsic enforcement of physical constraints, leading to limited generalization and poor accuracy on PDEs involving sharp gradients or shocks [238]. To reduce data reliance, unsupervised approaches like *Physics-Informed Neural Networks* (PINNs) incorporate PDE residuals directly into training losses [239], proving effective for elliptic and parabolic equations [240, 241]. However, PINNs encounter significant difficulties with hyperbolic PDEs, especially in capturing discontinuities and shock dynamics, resulting in unstable optimization, convergence failures, and inaccurate solutions [242, 243]. Recent variants, such as *Weak PINNs* (wPINNs) [244], *Parareal PINNs* (PPINNs) [245], and *Extended PINNs* (XPINNs) [246], aim to overcome these issues through weak formulations or specialized training strategies. Nonetheless, these adaptations often introduce considerable complexity and require extensive hyperparameter tuning, highlighting a persistent need for methods inherently suited to hyperbolic PDE challenges.

NNs for hyperbolic PDEs/conservation laws. Neural approaches tailored to hyperbolic PDEs have introduced innovations to handle shocks. Weak PINNs (wPINNs) [244] integrate weak-form residuals or integral constraints to mitigate issues with discontinuities.

Others employ neural networks directly within classical FV schemes to learn improved flux reconstructions [247, 248]. However, these enhancements typically reintroduce complexity, such as extensive manual parameterization or problem-specific adaptivity, diluting the key advantage of neural flexibility and generality.

Motivated by these limitations, our proposed NFV approach learns local update rules directly from data or PDE residuals. By preserving the fundamental conservation-law structure of traditional FV methods while flexibly leveraging neural networks, NFV achieves significantly higher accuracy, robustness, and scalability with minimal manual intervention.

8.3 Prerequisites and Notations: Finite Volume Methods

Standard FV methods, such as presented in [3], solve the integral form of the conservation law (8.1) on a mesh of uniform cells $I_i = [x_{i-1/2}, x_{i+1/2}]$, $i = 1, \cdots, I_{\max}$, with cell length Δx. The average of u over cell I_i at time $t_n = n\Delta t$, $n = 1, \cdots, N$, and the numerical flux through the interface $x_{i+1/2}$ over the time step, are given respectively by

$$u_i^n = \frac{1}{\Delta x} \int_{I_i} u(t_n, x)\, \mathrm{d}x \quad \text{and} \quad F_{i+1/2}^n = \int_{t_n}^{t_{n+1}} f(u(t, x_{i+1/2}))\, \mathrm{d}t, \tag{8.2}$$

where Δt is the time discretization. A first-order method $\mathcal{F}$ approximates the numerical flux as $\hat{F}_{i+1/2}^n = \mathcal{F}(u_i^n, u_{i+1}^n)$, while higher-order methods leverage additional cell averages.

Let us generalize this framework by including cell averages from previous time steps in order to construct even better approximations. Let FV_a^b be the class of methods that use a rectangular stencil of a neighboring spatial cells times b past time steps to estimate numerical fluxes. Specifically, let $\boldsymbol{u}_{i\pm1/2}^n(a-1, b)$ be the left and right $(a-1) \times b$ sub-stencils, as illustrated in blue and green in Figure 8.2. Then, an FV_a^b method $\mathcal{F}$ estimates numerical fluxes as $\hat{F}_{i\pm1/2}^n = \mathcal{F}(\boldsymbol{u}_{i\pm1/2}^n(a-1, b))$. Classical first-order methods, such as Godunov, fall under class FV_3^1; more details about their computation are provided in Appendix C.1. To our knowledge, the vast majority of FV methods in the literature use a single time step (i.e., $b = 1$) and a small number of spatial cells space cells. Indeed, designing analytical schemes with larger temporal or spatial stencils becomes exponentially more complex. Finally, the update rule is given by the exact relation

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x} \left(F_{i+1/2}^n - F_{i-1/2}^n \right), \tag{8.3}$$

which in practice is approximated using the numerical fluxes $\hat{F}_{i\pm1/2}^n$, leading to an approximation $\hat{u}_i^n$ of u_i^n. Note that the influx of one cell is the outflux of another, which ensures conservation.

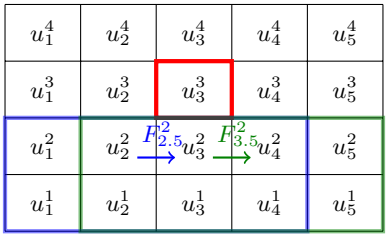


Figure 8.2: Example stencil for FV_5^2, which takes in a rectangular stencil of 2 time steps times 5 space cells to compute the next cell average (in red) using Equation (8.3) (specifically, both the in-flow and out-flow are computed using the illustrated 2x4 sub-stencils).

8.4 Our method: Neural Finite Volume (NFV)

Our method builds upon the FV framework by using neural networks to approximate the numerical flux. Specifically, we define NFV_a^b as a generalization of FV_a^b, where the numerical flux $\hat{F}^n_{i\pm 1/2}$ is predicted by a neural network $\mathcal{N}$ based on a local $a \times b$ spatiotemporal stencil:

$$\hat{F}^n_{i\pm 1/2} = \mathcal{N}(\boldsymbol{u}^n_{i\pm 1/2}(a-1, b))$$

The prediction of the solution is then updated using the classical FV update rule (8.3), ensuring mass conservation. We explore NFV models ranging from NFV_3^1 (matching Godunov's stencil) to NFV_{11}^{11}, using 11 spatial cells and 11 past time steps – configurations that would be exceedingly complex to design manually due to the high-dimensional stencil involved. This extension enables accurate learning even from noisy field data. In practice, we implement NFV as a CNN [236], which allows efficient computation across stencils due to the vectorized nature of CNNs. We refer to Appendix C.5 for implementation details.

We propose two variants: the supervised NFV_a^b, trained on solution data, and the unsupervised UNFV_a^b, trained directly from the PDE via a weak-form residual loss. Both share the same neural network architecture and only differ in their training objective. This framework is thus designed to be adapted to different kinds of problems.

8.4.1 Learning Objectives: Supervised and Unsupervised

Solving hyperbolic PDEs presents unique challenges, particularly due to the absence of closed-form solutions, even under simple initial conditions. To address this, we propose two distinct learning objectives: supervised and unsupervised. The supervised objective is applicable when reference solutions are available. Conversely, the unsupervised objective is employed when such solutions are unavailable, allowing the model to infer solutions by adhering to the underlying physical laws. Notably, supervised learning can also be utilized in scenarios where the governing equations are unknown, provided that observational data is accessible. This approach facilitates the application of numerical solvers to empirical data, imposing only fundamental physical constraints, such as mass conservation, without requiring extensive hyperparameter tuning (see Section 8.6).

8.4.1.1 Supervised Learning

Supervised learning offers a straightforward framework for training models when reference solutions are available. In this study, we employ supervised learning not only to approximate the solution of known equations but also to predict field data with unknown governing equations. Although solutions to hyperbolic PDEs are typically defined in the L_1 space, we consider their restrictions to compact subsets where the functions are bounded, thereby allowing treatment within the L_2 space. Accordingly, the loss function is defined as the standard mean square error:

$$\mathcal{L}_s = \mathop{\mathbb{E}}_{u_0 \sim \mathcal{R}} ||u - \hat{u}||_2^2$$

where u is the true solution, $\hat{u}$ is the predicted solution, and $\mathcal{R}$ is a distribution over initial conditions.

8.4.1.2 Unsupervised Learning

Unsupervised learning for hyperbolic PDEs is inherently more challenging due to the nature of their solutions. Closed-form solutions rarely exist, even for simple cases, and classical (strong) solutions often do not exist altogether. As a result, these equations are typically solved in a weak form. However, weak solutions are not unique, as multiple solutions can satisfy the PDE, but only one adheres to the entropy condition, making it physically meaningful.

The unsupervised loss function is defined to minimize the residuals of the weak formulation, in order to approximate the entropic solution. While imposing this loss does not guarantee convergence to the entropic solution, empirical results indicate that our method successfully converges to the entropy solution across various equations and numerous trials. To enhance learning efficiency, we optimize the weak formulation independently at each time step by minimizing the squared residuals. The collection of test functions Φ consists of 250 randomly sampled, compactly supported polynomials of degree 50 over the spatial domain. The unsupervised loss reads:

$$\mathcal{L}_w = \mathop{\mathbb{E}}_{\substack{\varphi \in \Phi \\ u_0 \sim \mathcal{R}}} \left[\sum_{n=1}^{N} \left(\sum_{i=1}^{I_{\max}} \left((\Delta t)^{-1} (\hat{u}_i^n - \hat{u}_i^{n-1}) \int_{I_i} \varphi + f(\hat{u}_i^n) [\varphi]_{x_{i-1/2}}^{x_{i+1/2}} \right) \right)^2 \right]$$

where $\hat{u}_i^n$ denotes the predicted solution at spatial index i and time step n, and $\mathcal{R}$ is a distribution over initial conditions.

8.5 Experiments

Experiments have been designed to answer four main questions:

- Is (U)NFV a compelling alternative to classical finite volume methods?
- Does UNFV converge to an entropic solution despite being trained on the weak formulation?

Table 8.1: Performance comparison between neural network models and classical numerical schemes. Results are computed over the evaluation set of 1000 piecewise constant initial conditions. For each method, we report mean and standard deviation in L_2 norm (mean$((u-\hat{u})^2)$).

	1st order FV					**Higher order FV**		**FEM**
	NFV$_3^1$	**UNFV$_3^1$**	GD	LF	EO	ENO	WENO	DG
G.shields	**1.3e$^{-4}_{\pm 4e^{-5}}$**	2.0e$^{-4}_{\pm 6e^{-5}}$	4.5e$^{-4}_{\pm 2e^{-4}}$	1.3e$^{-2}_{\pm 4e^{-3}}$	4.5e$^{-4}_{\pm 2e^{-4}}$	6.4e$^{-4}_{\pm 4e^{-4}}$	6.4e$^{-4}_{\pm 4e^{-4}}$	3.1e$^{-5}_{\pm 1e^{-5}}$
Tri. 1	**1.4e$^{-3}_{\pm 6e^{-4}}$**	1.9e$^{-3}_{\pm 9e^{-4}}$	2.3e$^{-3}_{\pm 1e^{-3}}$	9.6e$^{-3}_{\pm 4e^{-3}}$	2.3e$^{-3}_{\pm 1e^{-3}}$	2.0e$^{-3}_{\pm 2e^{-3}}$	1.9e$^{-3}_{\pm 2e^{-3}}$	2.6e$^{-4}_{\pm 1e^{-4}}$
Tri. 2	**2.4e$^{-3}_{\pm 1e^{-3}}$**	3.1e$^{-3}_{\pm 2e^{-3}}$	3.8e$^{-3}_{\pm 2e^{-3}}$	1.4e$^{-2}_{\pm 8e^{-3}}$	3.8e$^{-3}_{\pm 2e^{-3}}$	5.8e$^{-3}_{\pm 4e^{-3}}$	5.8e$^{-3}_{\pm 4e^{-3}}$	4.1e$^{-4}_{\pm 2e^{-4}}$
Trapez.	**1.1e$^{-3}_{\pm 4e^{-4}}$**	1.6e$^{-3}_{\pm 7e^{-4}}$	2.1e$^{-3}_{\pm 8e^{-4}}$	2.5e$^{-2}_{\pm 1e^{-2}}$	2.1e$^{-3}_{\pm 8e^{-4}}$	6.2e$^{-4}_{\pm 2e^{-4}}$	5.3e$^{-4}_{\pm 2e^{-4}}$	2.9e$^{-4}_{\pm 1e^{-4}}$
G.berg	**1.4e$^{-4}_{\pm 9e^{-5}}$**	3.8e$^{-4}_{\pm 2e^{-4}}$	4.9e$^{-4}_{\pm 2e^{-4}}$	5.3e$^{-3}_{\pm 2e^{-3}}$	4.9e$^{-4}_{\pm 2e^{-4}}$	1.1e$^{-3}_{\pm 6e^{-4}}$	1.2e$^{-3}_{\pm 9e^{-4}}$	3.4e$^{-4}_{\pm 2e^{-3}}$
U.wood	**3.8e$^{-4}_{\pm 1e^{-4}}$**	6.9e$^{-4}_{\pm 2e^{-4}}$	9.2e$^{-4}_{\pm 3e^{-4}}$	2.7e$^{-2}_{\pm 1e^{-2}}$	9.2e$^{-4}_{\pm 3e^{-4}}$	1.1e$^{-4}_{\pm 3e^{-5}}$	9.8e$^{-5}_{\pm 2e^{-5}}$	5.9e$^{-5}_{\pm 2e^{-5}}$
Burgers	**8.5e$^{-4}_{\pm 3e^{-4}}$**	1.3e$^{-3}_{\pm 6e^{-4}}$	1.9e$^{-3}_{\pm 7e^{-4}}$		2.6e$^{-3}_{\pm 1e^{-3}}$	2.7e$^{-3}_{\pm 1e^{-3}}$	2.8e$^{-3}_{\pm 1e^{-3}}$	1.0e$^{-4}_{\pm 4e^{-5}}$

- How does (U)NFV compare to much more complicated finite element methods?
- Can NFV perform well on field data that contains noise and may not be conservative?

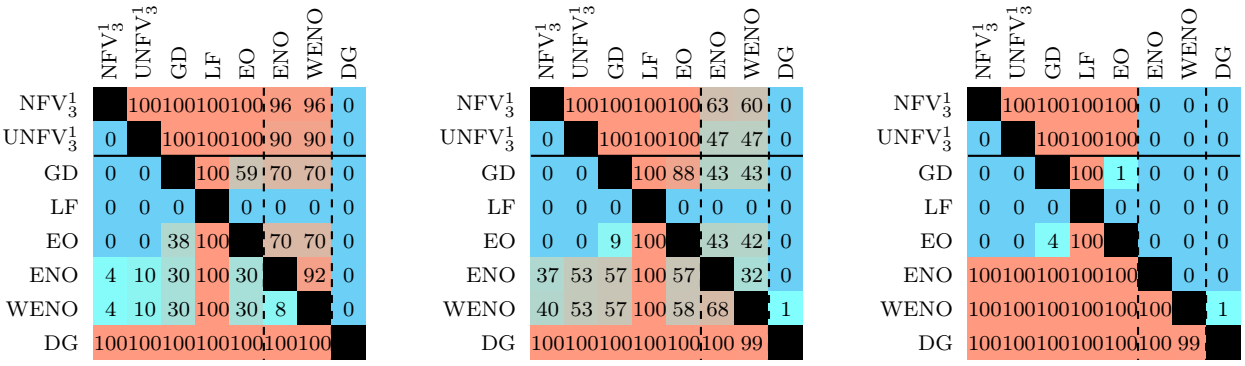
8.5.1 Baselines

Selecting appropriate baselines for PDE solvers poses challenges due to the diversity in computational frameworks: methods vary by mesh dependency (mesh-free versus mesh-based), solution generation (autoregressive versus single-pass), and generalizability (operator-based versus retrained per initial condition). Therefore, we adopt classical numerical schemes, the foundation of our NFV method, as baselines, ensuring a fair comparison. Given the fact that NFV is developed based on traditional first-order FV methods, the present work provides a compelling case for replacing standard FV solvers with the simpler yet effective NFV method whenever FV methods are typically employed. We consider all the numerical schemes introduced in Section 8.2 as baselines: first-order FV methods (Godunov, Lax-Friedrichs, and Engquist-Osher), higher-order ones (ENO, WENO), and DG, a finite-element method that is well-known for superior accuracy but suffers from computational burden. More details can be found in Appendix C.1.

8.5.2 Equations

The Lighthill-Whitham-Richards model [4, 5], known as LWR, is a first-order hyperbolic conservation law used to model traffic flow. It is expressed as

$$\partial_t \rho + \partial_x(\rho v(\rho)) = 0 \tag{8.4}$$



(a) Greenshields (b) Triangular (c) Underwood

Figure 8.3: **Comparison of numerical schemes across flow functions.** Each cell shows the proportion of the evaluation set on which the row scheme outperforms the column scheme. DG, the only FEM tested, is rarely beaten. NFV$_3^1$ and UNFV$_3^1$ outperform other first-order schemes and rival higher-order ones, making them strong choices depending on the equation.

ewhere ρ is the density of the traffic, $f : \rho \mapsto \rho v(\rho)$ is the flux function and v is the velocity. The flux function is typically modeled as a concave function of the density. Variations in the underlying velocity function give rise to different traffic flow models. In this chapter, six different models have been considered: Greenshields' [2], Triangular [249], Triangular skewed [249], Trapezoidal [250], Greenberg [251] and Underwood [252]. These models behave *very* differently and should be considered as different equations, as shown in Figure 8.1. Formulations and illustrations of those six models are given in Appendix C.2.

The inviscid Burgers' equation is a well-known hyperbolic conservation law used in various domains such as fluid mechanics [253], non-linear acoustics [254], gas dynamics [255], and traffic flow [256]. We refer the reader to [257] for a thorough introduction. It is expressed as

$$\partial_t u + \frac{1}{2}\partial_x u^2 = 0. \tag{8.5}$$

This equation can be written in the classical form of a conservation law using the flux function $f : u \mapsto 1/2 \cdot u^2$. Exact solutions to Riemann initial conditions are also known for this problem. Visualization of some solutions, including videos, are available on our webpage and in Figure 8.1.

While we focus on one-dimension equations in this chapter, our methods are, at least theoretically, generalizable to the multi-dimensional case of conservation laws [258]. In addition, we expect that our proposed methods can be extended to second-order hyperbolic equations such as ARZ-type models [259, 260] and shallow water equations [3]; see Appendix C.3 for details.

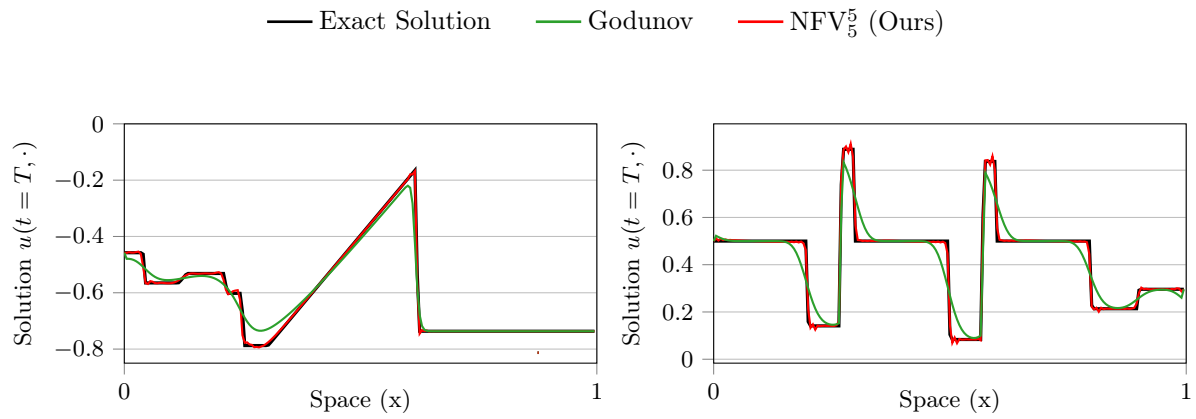


Figure 8.4: Comparison of the final density of the Burgers' equation (left) and LWR triangular equation (right) for NFV$_5^5$ and the Godunov Scheme. The proposed method displays an excellent approximation of the exact solution, capturing sharp features such as discontinuities and points of non-differentiability. It contains some minor oscillations in the solution, which are not present in the Godunov scheme. The latter, however, fails to capture the discontinuities and points of non-differentiability, offering a very smoothed solution.

8.5.3 Datasets

Training is performed using solutions derived from Riemann problems, which are initial value problems characterized by piecewise constant initial conditions with a single discontinuity (see Figure C.2 for examples). These problems are fundamental in the study of hyperbolic PDEs and serve as essential test cases for numerical methods. For the scenarios considered in this chapter, analytical solutions to Riemann problems are available, making supervised learning possible. Evaluation is performed on a more complicated set of a several hundred initial conditions to assess the model's generalization capabilities. These conditions consist of piecewise constant functions with ten discontinuities, introducing increased complexity. Exact solutions for these test cases are computed using the Lax-Hopf algorithm [225, 226, 227] on a finer grid (see Appendix C.5).

8.5.4 Results and discussion

Table 8.1 reports L_2 error for NFV$_3^1$, UNFV$_3^1$, and baseline methods across the seven benchmark equations. Our models consistently outperform all first-order FV methods, and surpass ENO/WENO schemes on about half of the equations. As expected, the higher-order DG method achieves significantly lower errors. Table 8.2 shows that NFV$_5^5$, while as simple to implement as standard NFV$_3^1$, achieves up to 10x better accuracy, approaching the performance of DG.

Figure 8.3 shows the fraction of test cases each method wins. NFV$_3^1$ and UNFV$_3^1$ consis-

Table 8.2: Evaluation of NFV_5^5 using piecewise constant initial conditions. Error is reported in L_2 norm. $\mathbf{NFV_5^5}$ achieve outstanding performance, gaining up to an order of magnitude improvement compared to Godunov and WENO. Its performance is close to DG, while keeping the implementation simplicity of a finite volume method and the computational complexity of NVF.

	Godunov	WENO	$\mathbf{NFV_3^1}$	$\mathbf{NFV_5^5}$	DG
Burgers'	$1.8\text{e}^{-3}{}_{\pm 6\text{e}^{-4}}$	$2.6\text{e}^{-3}{}_{\pm 1\text{e}^{-3}}$	$8.3\text{e}^{-4}{}_{\pm 3\text{e}^{-4}}$	$2.2\text{e}^{-4}{}_{\pm 1\text{e}^{-4}}$	$\mathbf{1.0\text{e}^{-4}}{}_{\pm 4\text{e}^{-5}}$
Greenshields	$4.1\text{e}^{-4}{}_{\pm 1\text{e}^{-4}}$	$6.9\text{e}^{-4}{}_{\pm 4\text{e}^{-4}}$	$1.2\text{e}^{-4}{}_{\pm 4\text{e}^{-5}}$	$4.6\text{e}^{-5}{}_{\pm 3\text{e}^{-5}}$	$\mathbf{4.2\text{e}^{-5}}{}_{\pm 2\text{e}^{-5}}$
Triangular	$2.2\text{e}^{-3}{}_{\pm 1\text{e}^{-3}}$	$2.0\text{e}^{-3}{}_{\pm 2\text{e}^{-3}}$	$1.3\text{e}^{-3}{}_{\pm 6\text{e}^{-4}}$	$2.9\text{e}^{-4}{}_{\pm 2\text{e}^{-4}}$	$\mathbf{2.7\text{e}^{-4}}{}_{\pm 1\text{e}^{-4}}$

tently surpass first-order FV methods. Against ENO/WENO, performance varies: our models outperform on some equations, match on others, and underperform in a few, highlighting the complexity of benchmarking across diverse problem settings. Still, the fact that NFV_3^1 and UNFV_3^1 consistently do better than first-order methods is seen as a sign that the approach appears to converge well. Specifically, NFV_3^1 and UNFV_3^1 consistently produce errors bounded by those of Godunov, emphasizing their robustness.

Since all methods use autoregressive prediction, evaluating performance at the final time step provides a good proxy for cumulative error. Figure 8.4 shows that the prediction of NFV_5^5 closely aligns with the exact solution, with only minor oscillations observed. Notably, NFV_5^5 effectively captures sharp discontinuities with high accuracy without relying on smoothing techniques, which are commonly employed in traditional FV methods to mitigate numerical artifacts.

A strong strength of classical numerical schemes is their proven convergence when the discretization is refined. While this is not proven for our method, Figure 8.5 shows that both NFV_3^1 and UNFV_3^1 can consistently perform better than Godunov, a scheme proven to converge, even when the grid is refined. Our schemes are thus expected to converge. The log-log plot exhibits a linear trends, which suggests that the rate of convergence could be polynomial.

8.6 Application: Modeling Large-Scale Experimental Field Data using NFV

We apply the proposed NFV method to large-scale traffic field data collected on Interstate 24 (I-24) in Tennessee, USA, using the I-24 MOTION infrastructure [180, 217], which enables high-resolution vehicle trajectory collection and constitutes the most extensive publicly available traffic dataset to date. Rather than predicting traffic speed, we focus on modeling traffic density, which is more directly tied to conservation laws and often exhibits sharp

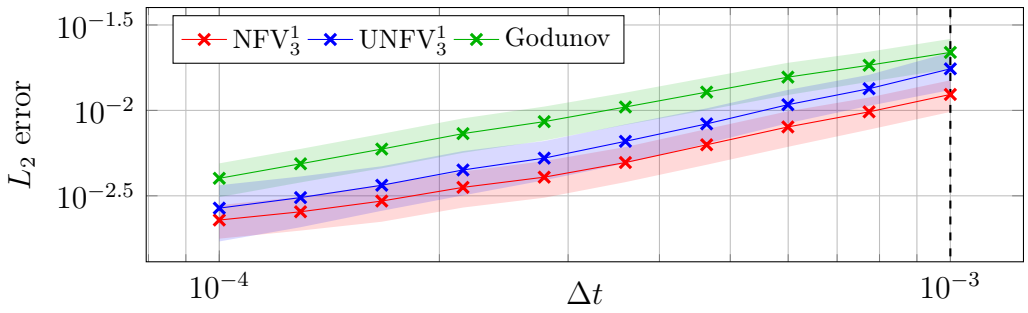


Figure 8.5: **Convergence plots on Greenshields' flux.** The L_2 error is computed against the exact solution on the evaluation set for different mesh discretizations. We report both error average and standard deviation, on a log-log scale. The dashed vertical line illustrates the discretization on which NFV_3^1 and UNFV_3^1 were trained; the models generalize to smaller discretizations. The ratio $\Delta t/\Delta x = 0.1$ remains constant as the mesh is refined.

transitions that are challenging to capture. Although conservation of mass is not strictly satisfied in highway traffic data due to merges, exits, and incidents, it serves as a strong inductive bias. We show that NFV achieves superior predictive accuracy compared to classical numerical schemes. Moreover, incorporating the PDE structure leads to substantially more stable training, particularly in data-scarce regimes. These findings suggest that our approach can enhance the accuracy and efficiency of traffic simulations, thereby contributing to better-informed decision-making in urban planning and traffic management.

8.6.1 Dataset

We evaluate our method on the I-24 MOTION dataset [180], which provides high-resolution vehicle trajectories collected on a four-mile stretch of Interstate 24 (mile markers 58.7 to 62.7) near Nashville, Tennessee. The data is captured by a network of high-definition cameras mounted along the highway as part of the I-24 MOTION infrastructure, leading to intricate trajectory data as illustrated in Figure C.3. Vehicle trajectories are reconstructed using a computer vision and data association pipeline [215], resulting in high-fidelity, though inherently noisy, field data.

The dataset consists of 10 days of vehicle trajectory data, collected during the morning rush hour (6:00 AM to 10:00 AM) over the 4-mile segment. From the raw trajectory data, we construct spatiotemporal vehicle density fields by aggregating vehicle counts over fixed spatial cells. Details of the data cleaning, processing, and preparation are provided in Appendix C.4.1. Visualization of the resulting density fields is shown in Figure C.4.

Table 8.3: **Improvements of NFV at different scales against numerical methods with fitted flow functions on field data.** The reported metrics include L1 error (mean($|u-\hat{u}|$)), L2 error (mean($(u-\hat{u})^2$)), and relative error (mean($|u-\hat{u}|/|\max\{\varepsilon, u\}|$)). The larger the input size of NFV, the better the performance. NFV_3^1 outperforms all calibrated Godunov fits, despite having the same input size and underlying structure.

	Calibrated numerical schemes (Godunov)					**NFV (Ours)**		
	Greenshields	Triangular	Trapezoidal	Greenberg	Underwood	NFV_3^1	NFV_5^5	NFV_{11}^{11}
L1	6.05e^{-2}	2.77e^{-2}	2.73e^{-2}	2.79e^{-2}	4.98e^{-2}	$\mathbf{2.37e^{-2}}$	2.31e^{-2}	$\mathbf{2.02e^{-2}}$
L2	1.93e^{-1}	1.31e^{-1}	1.30e^{-1}	1.33e^{-1}	1.81e^{-1}	$\mathbf{1.23e^{-1}}$	1.21e^{-1}	$\mathbf{1.09e^{-1}}$
Rel.	5.04e^{-1}	3.83e^{-1}	3.74e^{-1}	3.75e^{-1}	5.45e^{-1}	$\mathbf{3.57e^{-1}}$	3.51e^{-1}	$\mathbf{2.83e^{-1}}$

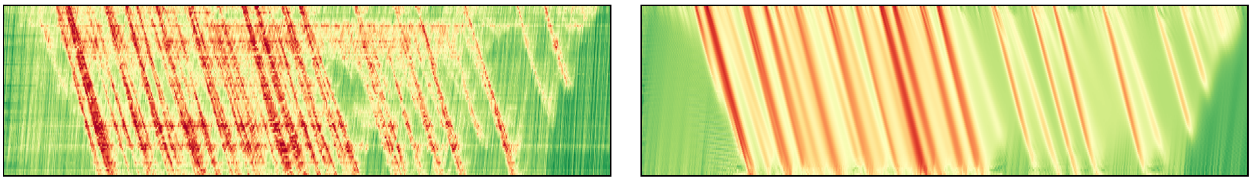


Figure 8.6: **Autoregressive prediction of NFV_{11}^{11} (right) compared to the ground truth (left).** Full results are shown in Figure C.7. See Appendix C.4.3 for how to read the heatmaps.

8.6.2 Results and Discussion

We compare NFV to numerical schemes using the flux functions from Appendix C.2. These functions, each defined by a few parameters, were calibrated via optimization to minimize the Godunov scheme's prediction error on the training set. The search ranges were intentionally broad, prioritizing predictive performance over physical plausibility to ensure a fair comparison. We evaluate three NFV variants of increasing capacity: NFV_3^1, NFV_5^5, and NFV_{11}^{11} (training details can be found in Appendix C.5), to assess how well they generalize and capture complex field dynamics.

Table 8.3 shows that all NFV models outperform the five tuned Godunov schemes, with performance improving as input size increases. This trend matches what was seen on synthetic data (Section 8.5). Despite training on just one hour of data, NFV predicts nearly four hours of traffic evolution autoregressively (Figure 8.6). While performance degrades in out-of-distribution zones (e.g., dark green regions unseen during training), the models still capture key wave patterns with high fidelity. Larger stencils help smooth out noise and improve accuracy, as seen in Figure C.7.

We further evaluate generalization on 7 other days. As shown in Figure C.8 and Table 8.4, NFV_{11}^{11} consistently outperforms the best Godunov scheme on the evaluation set, even though

both perform similarly on the training day. Indeed, although far from perfect, it is able to capture the evolution of free-flow traffic (dark green) with much greater accuracy, allowing it to successfully capture the end of congestion waves (red). NFV scales naturally with capacity: NFV_{11}^{11} adds only 1728 parameters over NFV_{3}^{1} but achieves significantly better accuracy with similar runtime and memory usage, unlike hand-crafted schemes, which significantly grow in complexity (see for example Appendix C.1).

Table 8.4: **Generalization of NFV against Godunov on 7 days of I-24 data never seen during training.** As in Table 8.3, we report mean and standard deviation of L1, L2 and relative errors.

	L1 error	L2 error	Relative error
Godunov	$1.56\text{e}^{-1} \pm 2.02\text{e}^{-2}$	$3.74\text{e}^{-2} \pm 8.25\text{e}^{-3}$	$6.26\text{e}^{-1} \pm 2.58\text{e}^{-1}$
NFV_{11}^{11}	$\mathbf{1.12\text{e}^{-1} \pm 7.39\text{e}^{-3}}$	$\mathbf{2.20\text{e}^{-2} \pm 2.59\text{e}^{-3}}$	$\mathbf{3.59\text{e}^{-1} \pm 7.58\text{e}^{-2}}$

8.7 Conclusion

We introduced a neural network-based framework for solving conservation laws that significantly improves accuracy over classical FV methods, while retaining the simplicity and structure of first-order schemes. Despite these empirical gains, a fundamental limitation remains: NFV and UNFV currently lack the rigorous theoretical guarantees, such as stability and convergence, established for traditional numerical methods. Nonetheless, the proposed approach demonstrates strong practical performance across diverse settings. Looking ahead, several important research directions warrant further exploration. First, establishing theoretical foundations for neural network-based solvers, particularly with respect to convergence and the preservation of physical constraints, is essential. Second, extending the methodology to more complex PDEs and high-dimensional domains will be critical for assessing its scalability and generalizability. Finally, hybrid strategies that integrate data-driven learning with classical numerical techniques may offer enhanced robustness and accuracy, bridging the gap between theoretical rigor and practical flexibility.

This work completes the methodological development that began with multi-agent reinforcement learning for traffic control, progressed through scalable simulation infrastructure and trajectory-based learning, and culminated in large-scale field deployment. The Neural Finite Volume methods developed here provide enhanced predictive capabilities that could further improve the performance of future autonomous vehicle traffic control systems. By combining physics-based conservation laws with data-driven learning, this approach represents a synthesis of the theoretical understanding and practical considerations that have guided the research program presented throughout this thesis. Together, these contributions demonstrate a complete pathway from algorithmic development to field validation, establishing

the feasibility and effectiveness of learning-based approaches for autonomous vehicle traffic control at scale.

More broadly, the proposed NFV framework has the potential to enhance simulations in domains such as traffic flow, environmental modeling, and fluid dynamics, thereby supporting more informed decision-making in urban planning and infrastructure development. However, reliance on data-driven models without rigorous validation may lead to inaccuracies, potentially resulting in misguided policies or designs. To mitigate such risks, we advocate for thorough validation against empirical data and collaboration with domain experts to ensure responsible application of the technology.

Chapter 9

Conclusion and Perspectives

This thesis began with a simple question: can a small number of autonomous vehicles smooth traffic flow and reduce energy consumption on highways? What started as simulation studies in idealized ring roads evolved into a complex journey of bridging the fundamental gap between controlled simulations and the messy reality of highway traffic.

The path from simulation to deployment revealed challenges I had not anticipated. Building fast, accurate simulations that faithfully reproduce the complex phenomena observed in highway traffic proved to be a demanding aspect of this work. Equally challenging was designing objective functions that could balance the competing demands of safety, comfort, energy efficiency, and throughput. This is a delicate optimization problem where small changes in reward formulation could lead to dramatically different learned behaviors.

But what was perhaps the most transformative was the opportunity to participate in the 100-vehicle field experiment on I-24. It is rare in academic research to take algorithms beyond the laboratory, especially at such scale and among human drivers navigating their daily commutes. Being part of this deployment, one of the largest autonomous vehicle traffic smoothing experiments ever conducted, was both humbling and exciting. It exposed countless practical considerations that simulation cannot capture, requiring extensive efforts to bridge the gap between controllers achieving excellent performance in simulation and making them work reliably on production vehicles. But at the end of this journey, we demonstrated that carefully designed learning-based controllers can indeed operate safely and beneficially in such complex environments.

The progression from early bottleneck control studies to the large-scale highway deployment fundamentally changed my understanding of what it means to develop autonomous vehicle algorithms for traffic applications. The most important insight may be that the traditional approach of perfecting algorithms in simulation before deployment might be insufficient. Instead, this work demonstrates the value of learning directly from trajectory data collected on the target highway, sidestepping the intractable problem of calibrating high-fidelity traffic microsimulators while maintaining the essential connection between simulation and reality.

The main methodological contribution is the development of learning-based traffic control systems that operate under realistic constraints: relying on local sensing, requiring no vehicle-

to-vehicle communication, and training directly on trajectory data rather than on calibrated microsimulators. The 100-vehicle field experiment on I-24 confirmed that decentralized control algorithms can meaningfully reduce the intensity of stop-and-go waves while maintaining safety standards. Additionally, the introduced neural finite volume methods demonstrated strong predictive performance on data from the I-24 highway, outperforming classical numerical schemes.

This work overall presents a complete pipeline, from algorithm development to large-scale deployment, supported by open-source tools and datasets that promote reproducible research in learning-based traffic control. The field validation highlights that, with careful system design and trajectory-based training, the gap between simulation and deployment in traffic applications can be effectively bridged. The following sections outline promising directions and open challenges for further advancing this line of research.

9.1 Faster and More Accurate Simulations

The trajectory-based training approach developed in this work demonstrated that using real highway data can improve wave dynamics accuracy compared to traditional microsimulators, though microsimulators certainly remain valuable for many applications. By replaying recorded I-24 trajectories, we captured complex wave phenomena without the difficult calibration process. However, there is still room for improvement: while lane changing models were developed for validation and evaluation purposes, most training happened in a single-lane fashion in the I-24 simulator, so dynamics accuracy could be further enhanced. Looking back, simulation speed is also something I should have focused on much earlier. Once the simulator is made faster through vectorization and GPU acceleration, the iterative process of training and tuning hyperparameters becomes significantly faster, enabling much more thorough exploration of the optimization space.

Recent data-driven simulators have pushed multi-agent traffic environments well beyond the limits of hand-tuned microsimulators in both speed and accuracy. Nocturne [57], presented in Chapter 4, runs at several thousand steps per second while giving each vehicle a human-like, occlusion-limited field of view and training on real Waymo trajectory data. This enables learning realistic driver models that capture human decision-making under partial observability, rather than relying on traditional rule-based models such as IDM. GPUDrive [261] expands Nocturne to natively run on GPU, surpassing one million steps per second. This corresponds to 2-3 orders of magnitude improvements over Nocturne, highlighting the importance of simulation optimization.

But fast simulators only pay off when the learner can keep up. SampleFactory [262], Madrona Engine [263] (which GPUDrive leverages), and PufferLib [264] all aim to accelerate reinforcement learning by building fast, parallel simulation environments through efficient architectures. They achieve this through various ways: C/C++ optimized code, efficient GPU operations with CUDA optimizations, highly-efficient multithreading for asynchronous environment sampling, unified wrappers to support many existing environments etc.

Overall there is a tradeoff between simulation speed and accuracy, but the research productivity gains from fast simulation environments cannot be overstated. While high-fidelity simulators that perfectly capture real-world dynamics remain valuable for final validation, fast simulators enable rapid iteration through the hyperparameter space, architecture exploration, and algorithm development that would be prohibitively expensive in slower environments. The ability to run thousands of experiments in parallel accelerates the pace of reinforcement learning research, allowing us to try out many ideas and converge on effective solutions faster. This suggests a staged approach, which has guided me during my PhD: rapid prototyping and hyperparameter search in fast simulators, followed by validation in higher-fidelity environments, to maximize both research speed and final performance.

9.2 Improved Traffic Prediction

The early simulation work in this thesis relied exclusively on local observations from vehicle sensors, achieving substantial traffic smoothing performance through purely decentralized control. However, the deployed I-24 controllers incorporated a fundamentally different sensing architecture. The hierarchical MegaController used a centralized speed planner that fused live probe data with vehicle telemetry to generate target speed profiles, providing macroscopic traffic information beyond what individual vehicles could observe locally.

While this macroscopic information expanded the available state space, the probe data also suffered from inherent limitations: high latency, temporal delays, and spatial aggregation over long highway segments. These inaccuracies constrain the planner's ability to provide precise, timely traffic state estimates that could enable more anticipatory control strategies.

Improved traffic prediction methods could substantially enhance controller performance. When controllers can predict the formation and propagation of stop-and-go waves, they can preemptively adjust vehicle behavior to absorb and dampen these disturbances before they fully develop. This anticipatory capability would represent a key advancement beyond purely reactive control approaches.

The Neural Finite Volume methods developed in Chapter 8 aim to address these prediction limitations. The goal of NFVM is to predict the future evolution of traffic states based on current measurements from sensors and other autonomous vehicles. By first estimating the local traffic state from available observations, NFVM can then predict how traffic density and flow patterns will evolve, including the formation, propagation, and dissipation of congestion waves.

This predictive capability could be enhanced by coupling physics-informed models with statistical patterns learned from historical data. Traffic congestion often exhibits regular temporal patterns during weekdays and seasonal variations, which could be incorporated into prediction frameworks to improve accuracy when no unusual events (such as accidents) occur. The integration of such models would require balancing physics-based wave propagation predictions with data-driven pattern recognition to handle both typical and anomalous traffic conditions.

9.3 Vehicle Communication

Explicit communication between autonomous vehicles represents a natural next step beyond the coordination strategies developed here. While our approach may exhibit some minor implicit coordination through traffic flow, direct vehicle-to-vehicle communication over 5G networks could enable more sophisticated cooperative behaviors. Expanding the action space to include explicit coordination signals, either with nearby vehicles or with all vehicles on the road, could lead to strategies that achieve higher throughput and energy efficiency.

While practical deployment would need to account for network latency and reliability constraints, a particularly interesting research question is quantifying the theoretical performance gains achievable under perfect communication conditions.

However, expanded action spaces and more complex interactions would also require advances in multi-agent reinforcement learning algorithms to maintain training stability. Current independent learning approaches may become insufficient as coordination complexity increases. Indeed, while the work presented in this thesis considered many agents operating together on the same road, the number of explicit interactions between autonomous vehicles remains limited, as most of the improvements are achieved through localized smoothing. However, when many agents require explicit communication, their actions heavily depend on each other's, making the problem highly coupled and thus more unstable and difficult to solve.

9.4 Hardware Integration

The practical deployment of learned traffic controllers requires deeper integration with vehicle hardware. Operating through stock adaptive cruise control systems, while providing a safe deployment pathway, imposes substantial limitations on control performance and response time. This approach introduces delays, restricts control to discrete increments, often fails at low speeds, and requires accurate ACC models for training, all of which adds a new layer of complexity during training. Direct acceleration control, as demonstrated in our limited field tests, provides substantially better performance but requires more comprehensive safety validation.

Developing safety frameworks for direct acceleration control involves ensuring that learned policies remain interpretable and their actions can be explained in safety-critical situations. The integration of generative AI and large language models could provide novel approaches to policy interpretation, offering natural language explanations of controller actions that could support both safety validation and regulatory approval processes.

The progression toward more capable hardware integration also requires advances in failsafe design and redundancy systems. As controllers gain more direct influence over vehicle dynamics, the consequences of system failures become more severe. Control architectures that maintain safety properties under sensor failures, communication losses, and model distribution shifts represent a necessary foundation for broader deployment.

9.5 Scaling Challenges

Moving from research validation to widespread adoption presents several interesting challenges, though the deployment pathway could be more straightforward than initially apparent. In an ideal scenario, since the controllers developped in this work operate through adaptive cruise control systems, they essentially function as improved ACC algorithms. Vehicle manufacturers could deploy these methods through software updates to existing ACC-equipped vehicles, leveraging the inherent safety guarantees of existing cruise control systems while allowing drivers to enable or disable the feature at any time.

However, user adoption presents a more subtle challenge. Driving according to the learned control policies may initially feel different to human drivers, as the controllers tend to open larger following gaps to absorb incoming traffic waves before they fully develop. While this behavior ultimately saves fuel and improves traffic flow for everyone, drivers might initially perceive it as less aggressive or responsive than their usual driving style. This suggests that user incentives will be important. Direct fuel savings for the individual driver, combined with broader societal benefits, could motivate adoption.

Integration with higher-level autonomous driving systems could sidestep the user experience issue entirely, as drivers would not directly feel the control adjustments. As autonomous vehicle penetration increases, the collective smoothing effects become more pronounced, creating a positive feedback loop for adoption.

Additionally, if manufacturers integrate the learned controllers directly into their vehicles, the reinforcement learning controller systems would gain access to the full suite of onboard sensors rather than relying on limited radar observations. This enhanced sensing could enable even stronger performance without necessarily requiring the macroscopic traffic state estimation systems used in our field experiments.

Policy and regulatory frameworks may need to evolve to address coordinated vehicle systems, particularly regarding system-level safety validation and liability allocation when multiple vehicles coordinate their actions. However, since the controllers maintain existing ACC safety properties, regulatory approval could follow established pathways for cruise control system updates rather than requiring entirely new frameworks for autonomous vehicle coordination.

Bibliography

[1] Lily Elefteriadou et al. *An introduction to traffic flow theory*. Vol. 84. Springer, 2014.

[2] Bruce D Greenshields, J Rowland Bibbins, WS Channing, and Harvey H Miller. "A study of traffic capacity". In: *Highway research board proceedings*. Vol. 14. 1. Washington, DC. 1935, pp. 448–477.

[3] Randall J LeVeque. *Finite volume methods for hyperbolic problems*. Vol. 31. Cambridge university press, 2002.

[4] Michael James Lighthill and G Be Whitham. "On kinematic waves I. Flood movement in long rivers". In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 229.1178 (1955), pp. 281–316.

[5] P. I. Richards. "Shock waves on the highway". In: *Operations Research* 4 (1956), pp. 42–51.

[6] Martin Treiber and Arne Kesting. *Traffic Flow Dynamics: Data, Models and Simulation*. Springer, 2013.

[7] Arne Kesting, Martin Treiber, and Dirk Helbing. "Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 368.1928 (2010), pp. 4585–4605.

[8] Peter G Gipps. "A behavioural car-following model for computer simulation". In: *Transportation research part B: methodological* 15.2 (1981), pp. 105–111.

[9] Yūki Sugiyama. "Optimal velocity model for traffic flow". In: *Computer Physics Communications* 121 (1999), pp. 399–401.

[10] Rui Jiang, Qingsong Wu, and Zuojin Zhu. "Full velocity difference model for a car-following theory". In: *Physical Review E* 64.1 (2001), p. 017101.

[11] Darbha Swaroop and J Karl Hedrick. "String stability of interconnected systems". In: *IEEE transactions on automatic control* 41.3 (2002), pp. 349–357.

[12] Shuo Feng, Yi Zhang, Shengbo Eben Li, Zhong Cao, Henry X Liu, and Li Li. "String stability for vehicular platoon control: Definitions and analysis methods". In: *Annual Reviews in Control* 47 (2019), pp. 81–97.

[13] Pablo Alvarez Lopez et al. "Microscopic Traffic Simulation using SUMO". In: *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. URL: https://elib.dlr.de/124092/.

[14] Cathy Wu, Aboudy Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M Bayen. "Flow: A Modular Learning Framework for Autonomy in Traffic". In: *arXiv preprint arXiv:1710.05465* (2017).

[15] Raphael E Stern et al. "Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments". In: *Transportation Research Part C: Emerging Technologies* 89 (2018), pp. 205–221.

[16] Richard Bellman. "Dynamic programming". In: *science* 153.3731 (1966), pp. 34–37.

[17] Edward Thorndike. *Animal intelligence: Experimental studies*. Routledge, 1911.

[18] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. Vol. 1. 1. MIT press Cambridge, 1998.

[19] Christopher JCH Watkins and Peter Dayan. "Q-learning". In: *Machine learning* 8.3 (1992), pp. 279–292.

[20] Richard S Sutton. "Learning to predict by the methods of temporal differences". In: *Machine learning* 3.1 (1988), pp. 9–44.

[21] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *nature* 518.7540 (2015), pp. 529–533.

[22] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3 (1992), pp. 229–256.

[23] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. "Trust Region Policy Optimization". In: *International Conference on Machine Learning (ICML)*. 2015.

[24] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[25] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[26] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), pp. 484–489.

[27] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[28] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International conference on machine learning*. Pmlr. 2018, pp. 1861–1870.

[29] Andrew Y Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. "Autonomous inverted helicopter flight via reinforcement learning". In: *Experimental robotics IX: The 9th international symposium on experimental robotics*. Springer. 2006, pp. 363–372.

[30] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. "End-to-end training of deep visuomotor policies". In: *Journal of Machine Learning Research* 17.39 (2016), pp. 1–40.

[31] Alex Kendall et al. "Learning to drive in a day". In: *2019 international conference on robotics and automation (ICRA)*. IEEE. 2019, pp. 8248–8254.

[32] Scott Fujimoto, Herke Van Hoof, and David Meger. "Addressing function approximation error in actor-critic methods". In: *arXiv preprint arXiv:1802.09477* (2018).

[33] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. "Asynchronous methods for deep reinforcement learning". In: *International conference on machine learning*. PMLR. 2016, pp. 1928–1937.

[34] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. "Offline reinforcement learning with implicit q-learning". In: *arXiv preprint arXiv:2110.06169* (2021).

[35] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. "Conservative q-learning for offline reinforcement learning". In: *Advances in neural information processing systems* 33 (2020), pp. 1179–1191.

[36] Zhihong Shao et al. *DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models*. 2024. arXiv: 2402.03300 [cs.CL]. URL: https://arxiv.org/abs/2402.03300.

[37] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. "Policy gradient methods for reinforcement learning with function approximation". In: *Advances in neural information processing systems* 12 (1999).

[38] Eugene Vinitsky, Nathan Lichtlé, Kanaad Parvate, and Alexandre Bayen. "Optimizing Mixed Autonomy Traffic Flow With Decentralized Autonomous Vehicles and Multi-Agent RL". In: *arXiv preprint arXiv:2011.00120* (2020).

[39] Fred L Hall and Kwaku Agyemang-Duah. "Freeway capacity drop and the definition of capacity". In: *Transportation research record* 1320 (1991).

[40] Koohong Chung, Jittichai Rudjanakanoknad, and Michael J Cassidy. "Relation between traffic density and capacity drop at three freeway bottlenecks". In: *Transportation Research Part B: Methodological* 41.1 (2007), pp. 82–95.

[41] Markos Papageorgiou, Habib Hadj-Salem, Jean-Marc Blosseville, et al. "ALINEA: A local feedback control law for on-ramp metering". In: *Transportation Research Record* 1320.1 (1991), pp. 58–67.

[42] Stef Smulders. "Control of freeway traffic flow by variable speed signs". In: *Transportation Research Part B: Methodological* 24.2 (1990), pp. 111–132.

[43] Yuankai Wu, Huachun Tan, and Bin Ran. "Differential variable speed limits control for freeway recurrent bottlenecks via deep reinforcement learning". In: *arXiv preprint arXiv:1810.10952* (2018).

[44] Xiao-Yun Lu, Pravin Varaiya, Roberto Horowitz, Dongyan Su, and Steven E Shladover. "A new approach for combined freeway variable speed limits and coordinated ramp metering". In: *13th International IEEE Conference on Intelligent Transportation Systems*. IEEE. 2010, pp. 491–498.

[45] Georgia-Roumpini Iordanidou, Claudio Roncoli, Ioannis Papamichail, and Markos Papageorgiou. "Feedback-based mainstream traffic flow control for multiple bottlenecks on motorways". In: *IEEE Transactions on Intelligent Transportation Systems* 16.2 (2014), pp. 610–621.

[46] Mladen Čičić, Li Jin, and Karl Henrik Johansson. "Coordinating Vehicle Platoons for Highway Bottleneck Decongestion and Throughput Improvement". In: *arXiv preprint arXiv:1907.13049* (2019).

[47] Richard Bellman. "A Markovian decision process". In: *Journal of Mathematics and Mechanics* (1957), pp. 679–684.

[48] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. "Multi-agent actor-critic for mixed cooperative-competitive environments". In: *Advances in neural information processing systems*. 2017, pp. 6379–6390.

[49] Joshua Achiam, Ethan Knight, and Pieter Abbeel. "Towards characterizing divergence in deep q-learning". In: *arXiv preprint arXiv:1903.08894* (2019).

[50] Stefan Krauß. "Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics". PhD thesis. 1998.

[51] Jakob Erdmann. "SUMO's lane-changing model". In: *Modeling Mobility with Open Data*. Springer, 2015, pp. 105–123.

[52] Jaime Barceló and Jordi Casas. "Dynamic network simulation with AIMSUN". In: *Simulation approaches in transportation analysis*. Springer, 2005, pp. 57–98.

[53] Eric Liang et al. "RLlib: Abstractions for distributed reinforcement learning". In: *International Conference on Machine Learning*. 2018, pp. 3053–3062.

[54] Anastasia D Spiliopoulou, Ioannis Papamichail, and Markos Papageorgiou. "Toll plaza merging traffic control for throughput maximization". In: *Journal of Transportation Engineering* 136.1 (2010), pp. 67–76.

[55] Eric Liang et al. *RLlib: Abstractions for Distributed Reinforcement Learning*. 2017. arXiv: 1712.09381 [cs.AI].

[56] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning". In: *arXiv preprint arXiv:1803.11485* (2018).

[57] Eugene Vinitsky, Nathan Lichtlé, Xiaomeng Yang, Brandon Amos, and Jakob Foerster. "Nocturne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world". In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 3962–3974. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/191e9e721a2748a860714fb23aaf7c5d-Paper-Datasets_and_Benchmarks.pdf.

[58] Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. ""Other-Play" for Zero-Shot Coordination". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 4399–4410.

[59] Julian Bernhard, Klemens Esterle, Patrick Hart, and Tobias Kessler. "BARK: Open behavior benchmarking in multi-agent environments". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 6201–6208.

[60] Ming Zhou et al. "Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving". In: *arXiv preprint arXiv:2010.09776* (2020).

[61] Quanyi Li, Zhenghao Peng, Zhenghai Xue, Qihang Zhang, and Bolei Zhou. "Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning". In: *arXiv preprint arXiv:2109.12674* (2021).

[62] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. "CARLA: An open urban driving simulator". In: *Conference on robot learning*. PMLR. 2017, pp. 1–16.

[63] Panpan Cai, Yiyuan Lee, Yuanfu Luo, and David Hsu. "Summit: A simulator for urban driving in massive mixed traffic". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 4023–4029.

[64] Praveen Palanisamy. "Multi-agent connected autonomous driving using deep reinforcement learning". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–7.

[65] Edouard Leurent. *An Environment for Autonomous Driving Decision-Making*. https://github.com/eleurent/highway-env. 2018.

[66] Matthias Müller, Vincent Casser, Jean Lahoud, Neil Smith, and Bernard Ghanem. "Sim4cv: A photo-realistic simulator for computer vision applications". In: *International Journal of Computer Vision* 126.9 (2018), pp. 902–919.

[67] Liam Paull et al. "Duckietown: an open, inexpensive and flexible platform for autonomy education and research". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 1497–1504.

[68] Anirban Santara, Sohan Rudra, Sree Aditya Buridi, Meha Kaushik, Abhishek Naik, Bharat Kaul, and Balaraman Ravindran. "Madras: Multi agent driving simulator". In: *Journal of Artificial Intelligence Research* 70 (2021), pp. 1517–1555.

[69] Parth Kothari, Christian Perone, Luca Bergamini, Alexandre Alahi, and Peter Ondruska. "Drivergym: Democratising reinforcement learning for autonomous driving". In: *arXiv preprint arXiv:2111.06889* (2021).

[70] Craig Quiter. *Deepdrive Zero*. Version alpha. 2020. DOI: 10.5281/zenodo.3871907. URL: https://doi.org/10.5281/zenodo.3871907.

[71] Alexander Amini, Tsun-Hsuan Wang, Igor Gilitschenski, Wilko Schwarting, Zhijian Liu, Song Han, Sertac Karaman, and Daniela Rus. "VISTA 2.0: An Open, Data-driven Simulator for Multimodal Sensing and Policy Learning for Autonomous Vehicles". In: *arXiv preprint arXiv:2111.12083* (2021).

[72] Nolan Bard et al. "The hanabi challenge: A new frontier for ai research". In: *Artificial Intelligence* 280 (2020), p. 103216.

[73] Mikayel Samvelyan et al. "The starcraft multi-agent challenge". In: *arXiv preprint arXiv:1902.04043* (2019).

[74] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. "The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games". In: *arXiv preprint arXiv:2103.01955* (2021).

[75] Joel Z Leibo et al. "Scalable evaluation of multi-agent reinforcement learning with melting pot". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 6187–6199.

[76] Scott Ettinger et al. "Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9710–9719.

[77] Benjamin Wilson et al. "Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting". In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. 2021.

[78] Holger Caesar et al. "nuscenes: A multimodal dataset for autonomous driving". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11621–11631.

[79] John Houston et al. "One thousand and one hours: Self-driving motion prediction dataset". In: *arXiv preprint arXiv:2006.14480* (2020).

[80] Wei Zhan et al. "Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps". In: *arXiv preprint arXiv:1910.03088* (2019).

[81] David Crundall, Geoffrey Underwood, and Peter Chapman. "Driving experience and the functional field of view". In: *Perception* 28.9 (1999), pp. 1075–1087.

[82] Herman Bouma. "Interaction effects in parafoveal letter recognition". In: *Nature* 226.5241 (1970), pp. 177–178.

[83] Yan Gu, Yong He, Kayvon Fatahalian, and Guy Blelloch. "Efficient BVH construction via approximate agglomerative clustering". In: *Proceedings of the 5th High-Performance Graphics Conference*. 2013, pp. 81–88.

[84] Jon Louis Bentley. *Decomposable searching problems*. Tech. rep. Carnegie-Mellon Univ Pittsburgh PA, Dept of Computer Science, 1978.

[85] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. "Vectornet: Encoding hd maps and agent dynamics from vectorized representation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11525–11533.

[86] Aleksei Petrenko, Zhehui Huang, Tushar Kumar, Gaurav Sukhatme, and Vladlen Koltun. "Sample factory: Egocentric 3d control from pixels at 100000 fps with asynchronous reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 7652–7662.

[87] Michael Bain and Claude Sammut. "A Framework for Behavioural Cloning." In: *Machine Intelligence 15*. 1995, pp. 103–129.

[88] Stefan Schaal. "Learning from demonstration". In: *Advances in neural information processing systems* 9 (1996).

[89] Maximilian Igl et al. "Symphony: Learning Realistic and Diverse Agents for Autonomous Driving Simulation". In: *arXiv preprint arXiv:2205.03195* (2022).

[90] Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. "Trafficsim: Learning to simulate realistic multi-agent behaviors". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 10400–10409.

[91] Avik Pal, Jonah Philion, Yuan-Hong Liao, and Sanja Fidler. "Emergent road rules in multi-agent driving environments". In: *arXiv preprint arXiv:2011.10753* (2020).

[92] Nathan Lichtlé, Kathy Jang, Adit Shah, Eugene Vinitsky, Jonathan W Lee, and Alexandre M Bayen. "Traffic smoothing controllers for autonomous vehicles using deep reinforcement learning and real-world trajectory data". In: *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2023, pp. 4346–4351.

[93] *Use of energy for transportation - U.S. Energy Information Administration (EIA) — eia.gov*. https://www.eia.gov/energyexplained/use-of-energy/transportation.php. [Accessed 04-Mar-2023].

[94] *U.S. energy facts explained - consumption and production - U.S. Energy Information Administration (EIA) — eia.gov*. https://www.eia.gov/energyexplained/us-energy-facts/. [Accessed 04-Mar-2023].

[95] *Highlights of the Automotive Trends Report | US EPA — epa.gov*. https://www.epa.gov/automotive-trends/highlights-automotive-trends-report. [Accessed 04-Mar-2023].

[96] *Autonomous Vehicles — content.naic.org*. https://content.naic.org/cipr-topics/autonomous-vehicles. [Accessed 04-Mar-2023].

[97] W Beaty. *Traffic "Experiments" and a Cure for Waves & Jams*. http://amasci.com/amateur/traffic/trafexp.html. [Accessed 15-Oct-2006]. 1998.

[98] Fangyu Wu et al. "Tracking vehicle trajectories and fuel rates in phantom traffic jams: Methodology and data". In: *Transportation Research Part C: Emerging Technologies* 99 (2019), pp. 82–109.

[99] Kathy Jang, Eugene Vinitsky, Behdad Chalaki, Ben Remer, Logan Beaver, Andreas Malikopoulos, and Alexandre Bayen. "Simulation to scaled city: zero-shot policy transfer for traffic control via autonomous vehicles". In: *2019 International Conference on Cyber-Physical Systems*. Montreal, CA, 2018. ISBN: 9781450362856.

[100] Eugene Vinitsky, Kanaad Parvate, Aboudy Kreidieh, Cathy Wu, and Alexandre Bayen. "Lagrangian control through deep-rl: Applications to bottleneck decongestion". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 759–765.

[101] Shumo Cui, Benjamin Seibold, Raphael Stern, and Daniel B Work. "Stabilizing traffic flow via a single autonomous vehicle: Possibilities and limitations". In: *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE. 2017, pp. 1336–1341.

[102] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates". In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE. 2017, pp. 3389–3396.

[103] Oriol Vinyals et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nature* 575.7782 (2019), pp. 350–354.

[104] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, et al. "Mastering the game of Go without human knowledge". In: *Nature* 550.7676 (2017), p. 354.

[105] Eugene Vinitsky et al. "Benchmarks for reinforcement learning in mixed-autonomy traffic". In: *Conference on Robot Learning*. PMLR. 2018, pp. 399–409.

[106] Yuki Sugiyama, Minoru Fukui, Macoto Kikuchi, Katsuya Hasebe, Akihiro Nakayama, Katsuhiro Nishinari, Shin-ichi Tadaki, and Satoshi Yukawa. "Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam". In: *New journal of physics* 10.3 (2008), p. 033001.

[107] Nathan Lichtlé, Eugene Vinitsky, Matthew Nice, Benjamin Seibold, Dan Work, and Alexandre M Bayen. "Deploying traffic smoothing cruise controllers learned from trajectory data". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 2884–2890.

[108] Matthew Nice et al. *The I-24 Trajectory Dataset*. Version 1.1.1. Zenodo, Sept. 2021. DOI: 10.5281/zenodo.6456348. URL: https://doi.org/10.5281/zenodo.6456348.

[109] Zhe Fu, Abdul Rahman Kreidieh, Han Wang, Jonathan W. Lee, Maria Laura Delle Monache, and Alexandre M. Bayen. *Cooperative Driving for Speed Harmonization in Mixed-Traffic Environments*. 2023. arXiv: 2302.07453 [eess.SY].

[110] Jonathan W. Lee et al. "Integrated Framework of Vehicle Dynamics, Instabilities, Energy Models, and Sparse Flow Smoothing Controllers". In: *Proceedings of the Workshop on Data-Driven and Intelligent Cyber-Physical Systems*. Nashville, TN, USA, 2021, pp. 41–47. ISBN: 9781450384452. DOI: 10.1145/3459609.3460530.

[111] Morris R Flynn, Aslan R Kasimov, J-C Nave, Rodolfo R Rosales, and Benjamin Seibold. "Self-sustained nonlinear waves in traffic flow". In: *Physical Review E* 79.5 (2009), p. 056113.

[112] I Ge Jin, Sergei S Avedisov, Chaozhe R He, Wubing B Qin, Mehdi Sadeghpour, and Gábor Orosz. "Experimental validation of connected automated vehicle design among human-driven vehicles". In: *Transportation research part C: emerging technologies* 91 (2018), pp. 335–352.

[113] Jiaqi Ma, Xiaopeng Li, Steven Shladover, Hesham A. Rakha, Xiao-Yun Lu, Ramanujan Jagannathan, and Daniel J. Dailey. "Freeway Speed Harmonization". In: *IEEE Transactions on Intelligent Vehicles* 1.1 (2016), pp. 78–89. DOI: 10.1109/TIV.2016.2551540.

[114] Michail Makridis, Konstantinos Mattas, Biagio Ciuffo, Fabrizio Re, Akos Kriston, Fabrizio Minarini, and Greger Rognelund. "Empirical study on the properties of adaptive cruise control systems and their impact on traffic flow and string stability". In: *Transportation research record* 2674.4 (2020), pp. 471–484.

[115] Victor L Knoop, Meng Wang, Isabel Wilmink, D Marika Hoedemaeker, Mark Maaskant, and Evert-Jeen Van der Meer. "Platoon of sae level-2 automated vehicles on public roads: Setup, traffic interactions, and stability". In: *Transportation Research Record* 2673.9 (2019), pp. 311–322.

[116] George Gunter et al. "Are commercially implemented adaptive cruise control systems string stable?" In: *IEEE Transactions on Intelligent Transportation Systems* (2020).

[117] Kathy Jang, Eugene Vinitsky, Behdad Chalaki, Ben Remer, Logan Beaver, Andreas A Malikopoulos, and Alexandre Bayen. "Simulation to scaled city: zero-shot policy transfer for traffic control via autonomous vehicles". In: *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*. 2019, pp. 291–300.

[118] Cathy Wu, Abdul Rahman Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M Bayen. "Flow: A Modular Learning Framework for Mixed Autonomy Traffic". In: *IEEE Transactions on Robotics* (2021).

[119] Jiaxun Cui, William Macke, Harel Yedidsion, Aastha Goyal, Daniel Urielli, and Peter Stone. "Scalable Multiagent Driving Policies For Reducing Traffic Congestion". In: *arXiv preprint arXiv:2103.00058* (2021).

[120] Behrad Toghi, Rodolfo Valiente, Dorsa Sadigh, Ramtin Pedarsani, and Yaser P Fallah. "Altruistic Maneuver Planning for Cooperative Autonomous Vehicles Using Multi-agent Advantage Actor-Critic". In: *arXiv preprint arXiv:2107.05664* (2021).

[121] Daniel A Lazar, Erdem Bıyık, Dorsa Sadigh, and Ramtin Pedarsani. "Learning how to dynamically route autonomous vehicles on shared roads". In: *Transportation Research Part C: Emerging Technologies* 130 (2021), p. 103258.

[122] Matthew Bunting, Rahul Bhadani, and Jonathan Sprinkle. "Libpanda: A High Performance Library for Vehicle Data Collection". In: *Proceedings of the Workshop on Data-Driven and Intelligent Cyber-Physical Systems*. 2021, pp. 32–40.

[123] Nice, M., Lichtle, N., Gumm, G., Roman, M., Vinitsky, E., Elmadani, S., and Bunting, M., Bhadani, R., Gunter,G., Kumar, M., and McQuade, S., Denaro, C., Delorenzo, R., Piccoli, B., Work, D. Bayen, A. Lee, J., Sprinkle, J. and Seibold, B. *The I-24 Trajectory Dataset*. doi.org/10.5281/zenodo.6366761. 2021.

[124] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[125] Safwan Elmadani, Matthew Nice, Matthew Bunting, Jonathan Sprinkle, and Rahul Bhadani. "From CAN to ROS: A Monitoring and Data Recording Bridge". In: *Proceedings of the Workshop on Data-Driven and Intelligent Cyber-Physical Systems*. 2021, pp. 17–21.

[126] Jonathan W. Lee et al. "Integrated Framework of Dynamics, Instabilities, Energy Models, and Sparse Flow Controllers". In: *Proceedings of the Workshop on CPS Data for Transportation and Smart cities with Human-in-the-loop*. 2021.

[127] Fang-Chieh Chou, Marsalis Gibson, Alexandre Bhadani Rahul and Bayen, and Jonathan Sprinkle. "Reachability Analysis for FollowerStopper: Safety Analysis and Experimental Results". In: *Proceedings of 2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021.

[128] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. *Stable Baselines3*. https://github.com/DLR-RM/stable-baselines3. 2019.

[129] Nathan Koenig and Andrew Howard. "Design and use paradigms for gazebo, an open-source multi-robot simulator". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. Vol. 3. IEEE. 2004, pp. 2149–2154.

[130] Matthew Nice, Safwan Elmadani, Rahul Bhadani, Matt Bunting, Jonathan Sprinkle, and Dan Work. "CAN coach: vehicular control through human cyber-physical systems". In: *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*. 2021, pp. 132–142.

[131] Jonathan W Lee et al. "Traffic control via connected and automated vehicles: An open-road field experiment with 100 cavs". In: *arXiv preprint arXiv:2402.17043* (2024).

[132] Kathy Jang et al. "Large-scale Wave Dampening via Single-Agent Reinforcement Learning Algorithms: Controlling and Deploying 100 Vehicles on a Highway with Learning-Based Algorithms". In: *IEEE Control Systems Magazine* (2024).

[133] William B Stevens. "The automated highway system program: A progress report". In: *IFAC Proceedings Volumes* 29.1 (1996), pp. 8180–8188.

[134] Han-Shue Tan, Rajesh Rajamani, and Wei-Bin Zhang. "Demonstration of an automated highway platoon system". In: *Proceedings of the 1998 American control conference. ACC (IEEE Cat. No. 98CH36207)*. Vol. 3. IEEE. 1998, pp. 1823–1827.

[135] Steven E Shladover, Christopher Nowakowski, Xiao-Yun Lu, and Robert Ferlis. "Cooperative adaptive cruise control: Definitions and operating concepts". In: *Transportation Research Record* 2489.1 (2015), pp. 145–152.

[136] DVAHG Swaroop, J Karl Hedrick, CC Chien, and Petros Ioannou. "A comparision of spacing and headway control laws for automatically controlled vehicles1". In: *Vehicle system dynamics* 23.1 (1994), pp. 597–625.

[137] Petros A Ioannou and Cheng-Chih Chien. "Autonomous intelligent cruise control". In: *IEEE Transactions on Vehicular technology* 42.4 (1993), pp. 657–672.

[138] Alain Girault. "A hybrid controller for autonomous vehicles driving on automated highways". In: *Transportation Research Part C: Emerging Technologies* 12.6 (2004), pp. 421–452. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2004.07.008. URL: https://www.sciencedirect.com/science/article/pii/S0968090X04000208.

[139] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The 2005 DARPA grand challenge: the great robot race*. Vol. 36. Springer, 2007.

[140] Sebastian Thrun et al. "Stanley: The robot that won the DARPA Grand Challenge". In: *Journal of field Robotics* 23.9 (2006), pp. 661–692.

[141] Lingyun Xiao and Feng Gao. "A comprehensive review of the development of adaptive cruise control systems". In: *Vehicle system dynamics* 48.10 (2010), pp. 1167–1192.

[142] Steven E Shladover, Christopher Nowakowski, Xiao-Yun Lu, and Raymond Hoogendoorn. "Using cooperative adaptive cruise control (CACC) to form high-performance vehicle streams". In: (2014).

[143] Arturo Dávila and Mario Nombela. *Sartre-safe road trains for the environment reducing fuel consumption through lower aerodynamic drag coefficient*. SAE International Warrendale, PA, USA, 2011.

[144] P Jootel. "Safe road trains for the environment (sartre) final report". In: *tech. rep., Technical Report for European Commission under the Framework 7 Programme Project 233683)* (2012).

[145] Sadayuki Tsugawa, Shin Kato, and Keiji Aoki. "An automated truck platoon for energy saving". In: *2011 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2011, pp. 4109–4114.

[146] Jeroen Ploeg, Steven Shladover, Henk Nijmeijer, and Nathan van de Wouw. "Introduction to the special issue on the 2011 grand cooperative driving challenge". In: *IEEE Transactions on Intelligent Transportation Systems* 13.3 (2012), pp. 989–993.

[147] Roozbeh Kianfar et al. "Design and experimental validation of a cooperative driving system in the grand cooperative driving challenge". In: *IEEE transactions on intelligent transportation systems* 13.3 (2012), pp. 994–1007.

[148] Cristofer Englund, Lei Chen, Jeroen Ploeg, Elham Semsar-Kazerooni, Alexey Voronov, Hoai Hoang Bengtsson, and Jonas Didoff. "The grand cooperative driving challenge 2016: boosting the introduction of cooperative automated vehicles". In: *IEEE Wireless Communications* 23.4 (2016), pp. 146–152.

[149] Steven Shladover, Xiao-Yun Lu, Shiyan Yang, Hani Ramezani, John Spring, Christopher Nowakowski, and David Nelson. "Cooperative adaptive cruise control (CACC) for partially automated truck platooning". In: (2018).

[150] Dehlia MC Willemsen, Antonius JC Schmeitz, and Edoardo Mascalchi. "EU ENSEMBLE Project: Specification of an interoperable solution for a support function for platooning". In: *IEEE Transactions on Intelligent Transportation Systems* (2023).

[151] Vicente Milanés, Steven E Shladover, John Spring, Christopher Nowakowski, Hiroshi Kawazoe, and Masahide Nakamura. "Cooperative adaptive cruise control in real traffic situations". In: *IEEE Transactions on intelligent transportation systems* 15.1 (2013), pp. 296–305.

[152] Vicente Milanés and Steven E Shladover. "Modeling cooperative and autonomous adaptive cruise control dynamic responses using experimental data". In: *Transportation Research Part C: Emerging Technologies* 48 (2014), pp. 285–300.

[153] Steven E Shladover, Dongyan Su, and Xiao-Yun Lu. "Impacts of cooperative adaptive cruise control on freeway traffic flow". In: *Transportation Research Record* 2324.1 (2012), pp. 63–70.

[154] Christopher Nowakowski, Jessica O'Connell, Steven E Shladover, and Delphine Cody. "Cooperative adaptive cruise control: Driver acceptance of following gap settings less than one second". In: *Proceedings of the human factors and ergonomics society annual meeting*. Vol. 54. 24. SAGE Publications Sage CA: Los Angeles, CA. 2010, pp. 2033–2037.

[155] Assad Al Alam, Ather Gattami, and Karl Henrik Johansson. "An experimental study on the fuel reduction potential of heavy duty vehicle platooning". In: *13th International IEEE Conference on Intelligent Transportation Systems*. IEEE. 2010, pp. 306–311.

[156] Stanley W Smith et al. "Improving urban traffic throughput with vehicle platooning: Theory and experiments". In: *IEEE Access* 8 (2020), pp. 141208–141223.

[157] Y. Sugiyama, M. Fukui, M. Kikuchi, K. Hasebe, A. Nakayama, K. Nishinari, S. Tadaki, and S. Yukawa. "Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam". In: *New Journal of Physics* 10.3 (Mar. 2008), p. 033001. DOI: 10.1088/1367-2630/10/3/033001. URL: https://dx.doi.org/10.1088/1367-2630/10/3/033001.

[158] C. Wu, A. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen. "Flow: Architecture and benchmarking for reinforcement learning in traffic control". In: *arXiv preprint arXiv:1710.05465* (2017), p. 10.

[159] Cathy Wu, Aboudy Kreidieh, Eugene Vinitsky, and Alexandre M. Bayen. "Emergent behaviors in mixed-autonomy traffic". In: *Conference on Robot Learning (CoRL)*. 2017. URL: http://proceedings.mlr.press/v78/wu17a.html.

[160] Jiaxun Cui, William Macke, Harel Yedidsion, Aastha Goyal, Daniel Urieli, and Peter Stone. "Scalable Multiagent Driving Policies For Reducing Traffic Congestion". In: *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. Virtual, May 2021.

[161] Yulin Zhang, William Macke, Jiaxun Cui, Daniel Urieli, and Peter Stone. *Learning a Robust Multiagent Driving Policy for Traffic Congestion Reduction*. 2023. arXiv: 2112.03759 [cs.AI].

[162] Cathy Wu, Alexandre M Bayen, and Ankur Mehta. "Stabilizing traffic with autonomous vehicles". In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 6012–6018.

[163] Eugene Vinitsky et al. "Benchmarks for reinforcement learning in mixed-autonomy traffic". In: *Proceedings of The 2nd Conference on Robot Learning*. Ed. by Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto. Vol. 87. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 399–409. URL: https://proceedings.mlr.press/v87/vinitsky18a.html.

[164] A. M. Bayen, J. W. Lee, B. Piccoli, B. Seibold, J. M. Sprinkle, and D. B. Work. "CIRCLES: Congestion Impacts Reduction via CAV-in-the-loop Lagrangian Energy Smoothing". In: Presented at the 2020 Vehicle Technologies Office Annual Merit Review, Washington, DC (virtual), 2020.

[165] A. M. Bayen, J. W. Lee, B. Piccoli, B. Seibold, J. M. Sprinkle, and D. B. Work. "CIRCLES: Congestion Impacts Reduction via CAV-in-the-loop Lagrangian Energy Smoothing". In: Presented at the 2021 Vehicle Technologies Office Annual Merit Review, Washington, DC (virtual), 2021.

[166] J. M. Sprinkle, A. M. Bayen, J. W. Lee, B. Piccoli, B. Seibold, and D. B. Work. "CIRCLES: Congestion Impacts Reduction via CAV-in-the-loop Lagrangian Energy Smoothing". In: Presented at the 2022 Vehicle Technologies Office Annual Merit Review, Washington, DC, 2022.

[167] Han Wang. "Centralized AV Speed Planner: Hierarchical Framework for Lagrangian Variable Speed Limit in Mixed Autonomy Traffic". In: *IEEE Control Systems Magazine* (2024).

[168] Nicolas Kardous et al. "A rigorous multi-population multi-lane hybrid traffic model for dissipation of waves via autonomous vehicles". In: *The European Physical Journal Special Topics* 231.9 (2022), pp. 1689–1700. DOI: 10.1140/epjs/s11734-022-00580-z. URL: https://doi.org/10.1140/epjs/s11734-022-00580-z.

[169] Andrea Bacciotti and Lionel Rosier. *Liapunov functions and stability in control theory*. Springer Science & Business Media, 2005.

[170] Kösaku Yosida. *Functional analysis*. Springer Science & Business Media, 2012.

[171] Alberto Bressan and Benedetto Piccoli. *Introduction to the mathematical theory of control*. Vol. 1. American institute of mathematical sciences Springfield, 2007.

[172] Eduardo Casas and Mariano Mateos. "Optimal control of partial differential equations". In: *Computational Mathematics, Numerical Analysis and Applications: Lecture Notes of the XVII'Jacques-Louis Lions' Spanish-French School* (2017), pp. 3–59.

[173] Lloyd N Trefethen. *Approximation Theory and Approximation Practice, Extended Edition*. SIAM, 2019.

[174] François Golse. "On the dynamics of large particle systems in the mean field limit". In: *Macroscopic and large scale phenomena: coarse graining, mean field limits and ergodicity* (2016), pp. 1–144.

[175] Paolo Baiti, Philippe G LeFloch, and Benedetto Piccoli. "Uniqueness of classical and nonclassical solutions for nonlinear hyperbolic systems". In: *Journal of Differential Equations* 172.1 (2001), pp. 59–82.

[176] Fangyu Wu and Alexandre M. Bayen. "A Hierarchical MPC Approach to Car-Following via Linearly Constrained Quadratic Programming". In: *IEEE Control Systems Letters* 7 (2023), pp. 532–537. DOI: 10.1109/LCSYS.2022.3201162.

[177] Zhe Fu, Abdul Rahman Kreidieh, Han Wang, Jonathan W Lee, Maria Laura Delle Monache, and Alexandre M Bayen. "Cooperative Driving for Speed Harmonization in Mixed-Traffic Environments". In: *2023 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2023, pp. 1–8.

[178] Mauro Garavello, Ke Han, and Benedetto Piccoli. *Models for vehicular traffic on networks*. Vol. 9. American Institute of Mathematical Sciences Springfield, MO, USA, 2016.

[179] Xiao-Yun Lu and Steven E Shladover. "Review of variable speed limits and advisories: Theory, algorithms, and practice". In: *Transportation research record* 2423.1 (2014), pp. 15–23.

[180] Derek Gloudemans et al. "I-24 MOTION: An instrument for freeway traffic science". In: *Transportation Research Part C: Emerging Technologies* 155 (2023), p. 104311.

[181] Derek Gloudemans, William Barbour, Nikki Gloudemans, Matthew Neuendorf, Brad Freeze, Said ElSaid, and Daniel B Work. "Interstate-24 motion: Closing the loop on smart mobility". In: *2020 IEEE Workshop on Design Automation for CPS and IoT (DESTION)*. IEEE. 2020, pp. 49–55.

[182] Alexandre Bayen, Maria Laura Delle Monache, Mauro Garavello, Paola Goatin, and Benedetto Piccoli. *Control problems for conservation laws with traffic applications: modeling, analysis, and numerical methods*. Springer Nature, 2022.

[183] Arwa Alanqary, Xiaoqian Gong, Alexander Keimer, Benjamin Seibold, Benedetto Piccoli, and Alexandre Bayen. "Optimal Control of Autonomous Vehicles for Flow Smoothing in Mixed Autonomy Traffic". In: *(to appear) 2023 IEEE 62nd Conference on Decision and Control (CDC)*. IEEE. 2023.

[184] R. Stern, D. Work, S. Cui, H. Pohlmann, B. Seibold, M. L. Delle Monache, B. Piccoli, and J. Sprinkle. "Stabilizing Traffic with a Single Autonomous Vehicle". In: *7th International Conference on Cyber-Physical Systems (ICCPS)*. ACM/IEEE. 2016.

[185] *Dissipation of Emergent Traffic Waves in Stop-and-Go Traffic Using a Supervisory Controller*. Vol. 57. Fontainbleau, Miami Beach, USA: IEEE, 2018, pp. 3628–3633. DOI: 10.1109/CDC.2018.8619700. URL: https://ieeexplore.ieee.org/document/8619700.

[186] Amaury Hayat et al. "A Holistic Approach to the Energy-Efficient Smoothing of Traffic via Autonomous Vehicles". In: *Intelligent Control and Smart Energy Management: Renewable Resources and Transportation*. Ed. by Maude Josée Blondin, João Pedro Fernandes Trovão, Hicham Chaoui, and Panos M. Pardalos. Cham: Springer International Publishing, 2022, pp. 285–316.

[187] John Baillieul and Panos J. Antsaklis. "Control and Communication Challenges in Networked Real-Time Systems". In: *Proceedings of the IEEE* 95.1 (2007), pp. 9–28. DOI: 10.1109/JPROC.2006.887290.

[188] Skylar Knickerbocker Vesal Ahsani Mostafa Amin-Naseri and Anuj Sharma. "Quantitative analysis of probe data characteristics: Coverage, speed bias and congestion detection precision". In: *Journal of Intelligent Transportation Systems* 23.2 (2019), pp. 103–119. DOI: 10.1080/15472450.2018.1502667. eprint: https://doi.org/10.1080/15472450.2018.1502667. URL: https://doi.org/10.1080/15472450.2018.1502667.

[189] Seoungbum Kim and Benjamin Coifman. "Comparing INRIX speed data against concurrent loop detector stations over several months". In: *Transportation Research Part C: Emerging Technologies* 49 (2014), pp. 59–72. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2014.10.002. URL: https://www.sciencedirect.com/science/article/pii/S0968090X14002940.

[190] Rahul Bhadani et al. "Approaches for Synthesis and Deployment of Controller Models on Automated Vehicles for Car-following in Mixed Autonomy". In: *Proceedings of Cyber-Physical Systems and Internet of Things Week 2023*. 2023, pp. 158–163.

[191] Matthew Nice, Matthew Bunting, Daniel Work, and Jonathan Sprinkle. "Middleware for a Heterogeneous CAV Fleet". In: *Proceedings of Cyber-Physical Systems and Internet of Things Week 2023*. 2023, pp. 86–91. URL: https://dl.acm.org/doi/abs/10.1145/3459609.3460531.

[192] Matthew Nice et al. "Parameter Estimation for Decoding Sensor Signals". In: *Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023)*. 2023, pp. 253–255.

[193] Matt Bunting, Rahul Bhadani, Matt Nice, Safwan Elmadani, and Jonathan Sprinkle. "Data from the Development Evolution of a Vehicle for Custom Control". In: *2022 2nd Workshop on Data-Driven and Intelligent Cyber-Physical Systems for Smart Cities Workshop (DI-CPS)*. IEEE. 2022, pp. 40–46.

[194] Mark Brackstone and Mike McDonald. "Car-following: a historical review". In: *Transportation Research Part F: Traffic Psychology and Behaviour* 2.4 (1999), pp. 181–196.

[195] Masako Bando, Katsuya Hasebe, Akihiro Nakayama, Akihiro Shibata, and Yuki Sugiyama. "Dynamical model of traffic congestion and numerical simulation". In: *Physical review E* 51.2 (1995), p. 1035.

[196] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. "Congested traffic states in empirical observations and microscopic simulations". In: *Physical review E* 62.2 (2000), p. 1805.

[197] George Gunter, Matthew Nice, Matt Bunting, Jonathan Sprinkle, and Daniel B Work. "Experimental testing of a control barrier function on an automated vehicle in live multi-lane traffic". In: *2022 2nd Workshop on Data-Driven and Intelligent Cyber-Physical Systems for Smart Cities Workshop (DI-CPS)*. IEEE. 2022, pp. 31–35.

[198] Alex Richardson, Matthew Walter Nice, Matt Bunting, Jonathan Lee, Rahul Bhadani, Daniel Work, and Jonathan Sprinkle. "Analysis of a Runtime Data Sharing Architecture over LTE for a Heterogeneous CAV Fleet". In: *Proceedings of Cyber-Physical Systems and Internet of Things Week 2023*. 2023, pp. 164–169.

[199] Saleh Albeaik et al. "Limitations and improvements of the intelligent driver model (IDM)". In: *SIAM Journal on Applied Dynamical Systems* 21.3 (2022), pp. 1862–1892.

[200] Xiaoqian Gong and Alexander Keimer. "On the well-posedness of the "Bando-follow the leader" car following model and a time-delayed version". In: *Networks and Heterogeneous Media* 18.2 (2023), pp. 775–798. DOI: 10.3934/nhm.2023033. URL: https://doi.org/10.3934/nhm.2023033.

[201] Amaury and Hayat. "Traffic Smoothing With a Single Vehicle: Dissipating Stop-and-Go Waves with a Single Controlled Vehicle in Dense Traffic: Experimental Evidence". In: *IEEE Control Systems Magazine* (2024).

[202] Argonne National Laboratory. *Autonomie*. https://www.anl.gov/taps/autonomie-vehicle-system-simulation-tool. 2023.

[203] Argonne National Laboratory (ANL). *Autonomie Compiled Vehicles. A technical manual for training in Autonomie*. Technical manual. Argonne National Laboratory. 2020.

[204] United States Environmental Protection Agency (US EPA). *Dynamometer Drive Schedules*. https://www.epa.gov/vehicle-and-fuel-emissions-testing/dynamometer-drive-schedules. Accessed: August 14, 2025. 2022.

[205] Nour Khoudari et al. "A Systematic Model Reduction Pipeline from Detailed Vehicle Energy Dynamics to Simple Models with Desirable Physics-like Properties". In: *IEEE Control Systems Magazine* (2024).

[206] Jonathan W. Lee et al. "Integrated Framework of Vehicle Dynamics, Instabilities, Energy Models, and Sparse Flow Smoothing Controllers". In: *Proceedings of the Workshop on Data-Driven and Intelligent Cyber-Physical Systems*. DI-CPS'21. Nashville, TN, USA: Association for Computing Machinery, 2021, pp. 41–47. ISBN: 9781450384452. DOI: 10.1145/3459609.3460530. URL: https://doi.org/10.1145/3459609.3460530.

[207] URL: https://comma.ai/.

[208] URL: https://intrepidcs.com.

[209] Rahul Bhadani, Matt Bunting, Matthew Nice, Ngoc Minh Tran, Safwan Elmadani, Dan Work, and Jonathan Sprinkle. "Strym: A python package for real-time can data logging, analysis and visualization to work with usb-can interface". In: *2022 2nd Workshop on Data-Driven and Intelligent Cyber-Physical Systems for Smart Cities Workshop (DI-CPS)*. IEEE. 2022, pp. 14–23. URL: https://ieeexplore.ieee.org/abstract/document/9805366.

[210] Paul Ngo and Jonathan Sprinkle. "Lightweight LSTM for CAN Signal Decoding". In: *Proceedings of the Workshop on Data-Driven and Intelligent Cyber-Physical Systems*. 2021, pp. 27–31.

[211] Paul Ngo, Jonathan Sprinkle, and Rahul Bhadani. "CANClassify: Automated Decoding and Labeling of CAN Bus Signals". In: *Berkeley Master's Thesis* 1.1 (2022), pp. 1–8.

[212] Rahul Bhadani and Jonathan Sprinkle. "Prototyping Vehicle Control Applications Using the CAT Vehicle Simulator". In: *arXiv preprint arXiv:2301.04574* (2023).

[213] Gergely Zachár. "Visualization of Large-Scale Trajectory Datasets". In: CPS-IoT Week '23. San Antonio, TX, USA: Association for Computing Machinery, 2023, pp. 152–157. ISBN: 9798400700491. DOI: 10.1145/3576914.3587710. URL: https://doi.org/10.1145/3576914.3587710.

[214] Derek Gloudemans and Daniel B Work. "Vehicle Tracking with Crop-based Detection". In: *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2021, pp. 312–319.

[215] Yanbing Wang, Derek Gloudemans, Zi Nean Teoh, Lisa Liu, Gergely Zachár, William Barbour, and Daniel Work. "Automatic vehicle trajectory data reconstruction at scale". In: *arXiv preprint arXiv:2212.07907* (2022).

[216] Derek Gloudemans, Yanbing Wang, Gracie Gumm, William Barbour, and Daniel B Work. "The Interstate-24 3D Dataset: a new benchmark for 3D multi-camera vehicle tracking". In: *arXiv preprint arXiv:2308.14833* (2023).

[217] Derek Gloudemans et al. "So you think you can track?" In: *arXiv preprint arXiv:2309.07268* (2023).

[218] Yanbing Wang, Junyi Ji, William Barbour, and Daniel B Work. "Online Min Cost Circulation for Multi-Object-Tracking on Fragments". In: *2023 IEEE International Intelligent Transportation Systems Conference (ITSC)*. To appear. IEEE. 2023.

[219] Mostafa Ameli et al. "Live Traffic Control Experiment via Automated Vehicles". In: *IEEE Control Systems Magazine* (2024).

[220] US Department of Transportation Federal Highway Administration. *Next Generation Simulation (NGSIM)*. Website. http://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm.

[221] Leslie C Edie. "Car-following and steady-state theory for noncongested traffic". In: *Operations research* 9.1 (1961), pp. 66–76.

[222] Nathan Lichtlé, Alexi Canesse, Zhe Fu, Hossein Nick Zinat Matin, Maria Laura Delle Monache, and Alexandre M Bayen. "(U)NFV: Supervised and Unsupervised Neural Finite Volume Methods for Solving Hyperbolic PDEs". In: *arXiv preprint arXiv:2505.23702* (2025).

[223] Alexi Canesse, Zhe Fu, Nathan Lichtlé, Hossein Nick Zinat Matin, Zihe Liu, Maria Laura Delle Monache, and Alexandre M Bayen. "A Tutorial on Neural Network-Based Solvers for Hyperbolic Conservation Laws: Supervised vs. Unsupervised Learning, and Applications to Traffic Modeling". In: (2025).

[224] Lawrence C Evans. *Partial differential equations*. Vol. 19. American Mathematical Society, 2022.

[225] PD Lax. "Hyperbolic systems of conservation laws II". In: *Communications on Pure and Applied Mathematics* 10.4 (1957), pp. 537–566.

[226] Christian G Claudel and Alexandre M Bayen. "Lax–Hopf based incorporation of internal boundary conditions into Hamilton–Jacobi equation. Part I: Theory". In: *IEEE Transactions on Automatic Control* 55.5 (2010), pp. 1142–1157.

[227] Christian G Claudel and Alexandre M Bayen. "Lax–hopf based incorporation of internal boundary conditions into hamilton-jacobi equation. part ii: Computational methods". In: *IEEE Transactions on Automatic Control* 55.5 (2010), pp. 1158–1174.

[228] Changqing Hu and Chi-Wang Shu. "A discontinuous Galerkin finite element method for Hamilton–Jacobi equations". In: *SIAM Journal on Scientific computing* 21.2 (1999), pp. 666–690.

[229] PD Lax. "The initial value problem for nonlinear hyperbolic equations in two independent variables". In: *Ann. Math. Studies* 33.21 (1954), pp. 1–229.

[230] SK Godunov. "A finite difference method for the computation of discontinuous solutions of the equations of fluid dynamics." In: *Sbornik: Mathematics* 47.8-9 (1959), pp. 357–393.

[231] Chi-Wang Shu. "High order ENO and WENO schemes for computational fluid dynamics". In: *High-order methods for computational physics*. Springer, 1999, pp. 439–582.

[232] Xu-Dong Liu, Stanley Osher, and Tony Chan. "Weighted essentially non-oscillatory schemes". In: *Journal of computational physics* 115.1 (1994), pp. 200–212.

[233] Bernardo Cockburn and Chi-Wang Shu. "The local discontinuous Galerkin method for time-dependent convection-diffusion systems". In: *SIAM Journal on Numerical Analysis* 35.6 (1998), pp. 2440–2463.

[234] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. "Fourier neural operator for parametric partial differential equations". In: *arXiv preprint arXiv:2010.08895* (2020).

[235] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators". In: *Nature machine intelligence* 3.3 (2021), pp. 218–229.

[236] Yann LeCun, Yoshua Bengio, et al. "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.

[237] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. "Geometric deep learning: going beyond euclidean data". In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[238] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. "Characterizing possible failure modes in physics-informed neural networks". In: *Advances in neural information processing systems* 34 (2021), pp. 26548–26560.

[239] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations". In: *arXiv preprint arXiv:1711.10561* (2017).

[240] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational physics* 378 (2019), pp. 686–707.

[241] Ameya D Jagtap, Ehsan Kharazmi, and George Em Karniadakis. "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems". In: *Computer Methods in Applied Mechanics and Engineering* 365 (2020), p. 113028.

[242] Feng Wang and Huaping Liu. "Understanding the behaviour of contrastive loss". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 2495–2504.

[243] Olga Fuks and Hamdi A Tchelepi. "Limitations of physics informed machine learning for nonlinear two-phase transport in porous media". In: *Journal of Machine Learning for Modeling and Computing* 1.1 (2020).

[244] Tim De Ryck, Siddhartha Mishra, and Roberto Molinaro. "wPINNs: Weak physics informed neural networks for approximating entropy solutions of hyperbolic conservation laws". In: *SIAM Journal on Numerical Analysis* 62.2 (2024), pp. 811–841.

[245] Xuhui Meng, Zhen Li, Dongkun Zhang, and George Em Karniadakis. "PPINN: Parareal physics-informed neural network for time-dependent PDEs". In: *Computer Methods in Applied Mechanics and Engineering* 370 (2020), p. 113250.

[246] Ameya D Jagtap and George Em Karniadakis. "Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations". In: *Communications in Computational Physics* 28.5 (2020).

[247] Tatiana Kossaczká, Matthias Ehrhardt, and Michael Günther. "Enhanced fifth order WENO shock-capturing schemes with deep learning". In: *Results in Applied Mathematics* 12 (2021), p. 100201.

[248] Yunjin Tong, Shiying Xiong, Xingzhe He, Shuqi Yang, Zhecheng Wang, Rui Tao, Runze Liu, and Bo Zhu. "RoeNet: Predicting discontinuity of hyperbolic systems from continuous data". In: *International Journal for Numerical Methods in Engineering* 125.6 (2024), e7406.

[249] Nikolas Geroliminis and Carlos F Daganzo. "Existence of urban-scale macroscopic fundamental diagrams: Some experimental findings". In: *Transportation Research Part B: Methodological* 42.9 (2008), pp. 759–770.

[250] Nikolas Geroliminis and Jie Sun. "Properties of a well-defined macroscopic fundamental diagram for urban traffic". In: *Transportation Research Part B: Methodological* 45.3 (2011), pp. 605–617.

[251] Harold Greenberg. "An analysis of traffic flow". In: *Operations research* 7.1 (1959), pp. 79–85.

[252] Robin T Underwood. "Speed, volume and density relationships". In: *Quality and theory of traffic flow* (1961).

[253] JM Burgers. "Math. examples illustrating relations occuring in the theory of turbulent fluid motion. Akademie van Wetenschappen". In: *Amsterdam. 1. Sect.* 17 (1939), p. 2.

[254] Bruno Lombard, Denis Matignon, and Yann Le Gorrec. "A fractional Burgers equation arising in nonlinear acoustics: theory and numerics". In: *IFAC Proceedings Volumes* 46.23 (2013), pp. 406–411.

[255] DE Panayotounakos and D Drikakis. "On the closed-form solutions of the wave, diffusion and Burgers equations in fluid mechanics". In: *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik* 75.6 (1995), pp. 437–447.

[256] Toshimitsu Musha and Hideyo Higuchi. "Traffic current fluctuation and the Burgers equation". In: *Japanese journal of applied physics* 17.5 (1978), p. 811.

[257] MARIA Cameron. "Notes on the Burgers Equation". In: *University of Marylan* (2011).

[258] A. Bressan. "Leggi di conservazione". In: *La Matematica nella Societa e nella Cultura* (2003).

[259] AATM Aw and Michel Rascle. "Resurrection of" second order" models of traffic flow". In: *SIAM journal on applied mathematics* 60.3 (2000), pp. 916–938.

[260] HM Zhang. "Structural properties of solutions arising from a nonequilibrium traffic flow theory". In: *Transportation Research Part B: Methodological* 34.7 (2000), pp. 583–603.

[261] Saman Kazemkhani, Aarav Pandya, Daphne Cornelisse, Brennan Shacklett, and Eugene Vinitsky. "GPUDrive: Data-driven, multi-agent driving simulation at 1 million FPS". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2025. arXiv: 2408.01584 [cs.AI]. URL: https://arxiv.org/abs/2408.01584.

[262] Aleksei Petrenko, Zhehui Huang, Tushar Kumar, Gaurav S. Sukhatme, and Vladlen Koltun. "Sample Factory: Egocentric 3D Control from Pixels at 100000 FPS with Asynchronous Reinforcement Learning". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 7652–7662. URL: http://proceedings.mlr.press/v119/petrenko20a.html.

[263] Brennan Shacklett et al. "An Extensible, Data-Oriented Architecture for High-Performance, Many-World Simulation". In: *ACM Trans. Graph.* 42.4 (2023).

[264] Joseph Suarez. *PufferLib: Making Reinforcement Learning Libraries and Environments Play Nice*. 2024.

[265] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. "Prioritized experience replay". In: *arXiv preprint arXiv:1511.05952* (2015).

[266] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[267] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078* (2014).

[268] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[269] S. K. Godunov. "A finite difference method for the numerical computation of discontinuous solutions of the equations of fluid dynamics". In: *Mathematicheckii Sbornik* 47 (1959), pp. 271–290.

[270] Björn Engquist and Stanley Osher. "One-sided difference approximations for nonlinear conservation laws". In: *Mathematics of Computation* 36.154 (1981), pp. 321–351.

[271] A. Aw and M. Rascle. "Resurrection of "second order" models of traffic flow". In: *SIAM Journal on Applied Mathematics* 60 (2000), pp. 916–938.

[272] H Michael Zhang. "A non-equilibrium traffic model devoid of gas-like behavior". In: *Transportation Research Part B: Methodological* 36.3 (2002), pp. 275–290.

[273] Geoff Langdale and Daniel Lemire. "Parsing gigabytes of JSON per second". In: *The VLDB Journal* 28.6 (2019), pp. 941–960.

Appendix A

Bottleneck Control

A.1 Training Parameters

In Table A.1, we outline the optimal hyperparameters and seed for every experiment presented in this chapter. These hyperparameters were found, as discussed in Sec. 3.3.5, by sweeping a fixed set of hyperparameters, picking the policy with the highest reward after 2000 iterations, and then sweeping 35 seeds.

A.2 Experiment Details

For the training parameters for TD3, we primarily used the default parameters set in RLlib [55][1] version 0.8.0, a distributed deep RL library. Both the policy and the Q-function are approximated by a neural network, each with two hidden layers of size $[400, 300]$ and a ReLU non-linearity following each hidden layer. We used a training buffer size of 100000 samples and use a ratio of 5 new samples from the environment for every gradient step. For each training run, we also perform a hyperparameter sweep over the following values:

- The learning rate for both the policy and the critic: $[1e-3, 1e-4]$.
- The length of the reward sequence before truncating the target with the Q-function (also known as *n-step return*): $[1, 10]$
- We test both using and not using prioritized experience replay [265].

The best performing value, in terms of final converged reward, is selected from the hyperparameters, after what we run 35 random seeds using the best hyperparameters. We select the highest reward at the end of training from these random seeds.

The percentage of autonomous vehicles varies among 5%, 10%, 20% and 40%. During each training rollout, we keep a fixed inflow of 2400 vehicles per hour over the whole horizon.

[1] https://github.com/ray-project/ray/python/ray/rllib

Table A.1: Parameters used during training with RLlib [53]'s implementation of the TD3 algorithm for the experiments trained at penetrations of 5%, 10%, 20%, 40% or universally (cf. Sec. 3.4.2) for the minimal, minimal + aggregate and radar + aggregate state spaces. The experiments trained at fixed penetration have been trained for 2000 iterations with a parameter search (cf. Sec A.2) followed by a grid search on the best parameters (cf. Sec A.3), after which the best seed was kept. The universal controllers have been trained for 400, 1200 and 2000 iterations for respectively the minimal, minimal + aggregate and radar + aggregate state spaces, and no seed search was ran for these three experiments. The last line of the table refers to the experiment that was trained without macroscopic information about the bottleneck's outflow (cf. Sec 3.4.3); it was trained for 1600 iterations and without seed search. Only parameters that differ from the default RLlib configuration for TD3 (https://docs.ray.io/en/releases-0.6.6/rllib-algorithms.html#deep-deterministic-policy-gradients-ddpg-td3) are detailed here.

		actor_lr	critic_lr	n_step	prioritized_replay	seed
minimal	5%	0.001	0.0001	5	True	24
	10%	0.0001	0.0001	5	False	29
	20%	0.0001	0.001	5	True	15
	40%	0.0001	0.0001	5	False	9
	universal	0.0001	0.0001	5	False	None
minimal + aggregate	5%	0.0001	0.0001	5	False	28
	10%	0.001	0.0001	5	True	3
	20%	0.0001	0.0001	5	False	0
	40%	0.0001	0.001	5	True	9
	universal	0.001	0.001	5	False	None
radar + aggregate	5%	0.0001	0.0001	5	True	14
	10%	0.0001	0.001	5	False	16
	20%	0.0001	0.001	5	False	17
	40%	0.0001	0.001	5	True	19
	universal	0.0001	0.0001	5	False	None
no congest info	5%	0.0001	0.001	5	False	None

At each time-step, a random number of vehicles are emitted from the start edge. Thus, the number of vehicles in each platoon behind the AVs will be of variable length and it is possible that at any time-step any given lane may have zero AVs in it. To populate the simulation fully with vehicles, we allow the experiment to run uncontrolled for 300 seconds. After that, the horizon is set to 1000 more seconds.

At training time, we use the re-routing technique discussed in Sec. 3.3.4 where vehicles are simply placed back at the beginning of the network after exiting. However, when performing the inflow-outflow sweeps to evaluate the efficacy of the policy / generate the graphs in this chapter, we turn rerouting off to ensure that our policy's performance is not dependent on the policy using the rerouting to generate some unusual behavior. To compute the outflow at a given inflow value, we run the system for 1000 seconds and compute the outflow over the last 500 seconds.

We use the traffic micro-simulator SUMO [13] for running our simulations. We use a simulation step of 0.5 seconds and a first-order Euler integration for the dynamics. While we use a relatively small time-step to maintain sensible dynamics, we use action repetition and only select a new controller action every 2.5 seconds. Each action is thus repeated five times; this approach is useful for speeding up training when the system dynamics change at a slower time-scale than the dynamics update frequency. This technique is standard when applying deep RL to Atari games [266]. Similarly, states, rewards and actions are only computed once every 5 simulation steps during both training and evaluation. Thus, running the system for 1000 seconds corresponds to 400 environment steps but to 2000 simulation steps.

It is essential to note that for the multi-agent experiments we used a *shared controller*, all of the agents operate in a decentralized fashion but share the same controller.

A.3 Reproducibility and Experimental Details

The code used to run the experiments and plot all of the figures is available at our fork of `Flow`[2].

For each example, we perform the hyperparameter sweep indicated in Sec. 3.3.5 and train at a fixed inflow of 2400 vehicles per hour. We then take the best hyperparameter set and re-run the experiment using 35 different seeds. The seed with the highest outflow is taken as the controller for each example.

All experiments are run on c4.8xlarge machines on AWS EC2 which have 36 virtual cores each. Since we use a single-processor implementation of TD3, both our hyperparameter sweeps and seed sweeps fit on a single machine.

A.4 Feedback Controller Sweep Parameters

For our feedback controllers, we swept the following hyperparameters in a grid:

[2] https://github.com/eugenevinitsky/decentralized_bottlenecks

- $n_{\text{crit}} = [6, 8, 10]$
- $K = [1, 5, 10, 20, 50]$
- $q_{\text{init}} = [200, 600, 1000, 5000, 10000]$

Empirically, we found that these were the parameters that the control scheme was most sensitive to. Whenever a vehicle enters, we set $q_0 = q_{\text{init}}$ so each vehicle is maintaining its own counter of the appropriate wait-time.

Appendix B

Nocturne

B.1 Experiment Details

B.1.1 Architecture and Computational Resources

The APPO experiments are each run for two days on 1 V100 GPU and 10 CPUs (Intel Xeon CPU E5-2698 v4 @ 2.20GHz) and each experiment is run for 5 seeds. For the RL experiments we run 5 conditions over 5 seeds for 2 days leading to a total of 1200 GPU hours and 12000 CPU hours. For the Imitation Learning experiments we run 5 seeds for three hours leading to 15 GPU hours and 100 CPU hours.

The architecture for the APPO agent is a two-layer neural network with 256 hidden units and Tanh non-linearities followed by a Gated Recurrent Unit [267] with 256 hidden states. The Behavior Cloning experiment is run on a single GPU for five seeds, trained on 1000 files, for 600 gradient steps with a batch size of 512. Here instead of using a recurrent neural network we stack the current state and the prior 4 states as an input and pass this through a three-layer (1025, 256, 128) neural network that then outputs two categorical heads, one for acceleration and one for steering. The acceleration head is binned into 15 bins equally spaced between $[-6, 6]\,\frac{\mathrm{m}}{\mathrm{s}^2}$ and steering to 43 equally spaced bins between $[-0.7, -0.7]$ radians. As there is no corresponding head tilt in the expert actions we extend the angle of the visibility cone to π radians. Note that this violates a rule on the cone shape set forth in Sec. 4.3.4; figuring out how to add head tilt to imitation agents remains an open question.

B.1.2 Hyperparameters

For APPO we use almost all the default hyperparameters of SampleFactory [86] at commit aed6cc92a7eb3510c4d4bcfac083ced07b5222f9. However, we use a batch size of 7168 and scale the observations by 10.0.

For Imitation Learning we use a learning rate of $3 \cdot 10^{-4}$, a batch size of 512, and stack a total of 5 states together to endow the agents with memory.

B.1.3 State Space Details

Since likely control architectures require the vector to be of a fixed size, we select a subset of visible road points and objects if there are too many and pad with a vector of 0.0's if there are too few. We sort both the road points, objects, and stop signs by distance and return the $500, 16, 4$ closest ones respectively where all three values are configurable by user. Using these values, the total state dimensionality is 6727. We note that this is a fairly high dimensional state and intentionally do not prune it to be smaller as dealing with this high dimensional input is a meaningful part of the benchmark. However, since we expect that users will likely want to construct their own types of states, utility functions are also provided to allow users to query the set of observed road points and objects.

B.1.4 Features

The features of the ego object are:

- The speed of the object.
- The distance and angle to the goal.
- Its width and length.
- The relative speed and heading to the target speed and heading.

Road object features are:

- The speed of the object.
- The angle between the velocity vectors of the object and of the ego vehicle.
- Its width and length.
- The angle and distance to the road object's position relative to the ego vehicle.
- Its heading relative to the ego vehicle.
- A one-hot vector indicating whether the object is a vehicle, pedestrian, or cyclist.

As per the VectorNet representation, each road point is part of a discretized polyline with an approximate discretization size of 0.5 meters. To ensure that any vehicle is able to discern which road points are connected to each other, we include a vector pointing to the neighbor of the point in the polyline. As with the road object representation, all points are in the frame of the observing vehicle. Road point features are:

- The angle and distance to the road point's position relative to the ego vehicle.
- A 2D element representing the vector pointing from the current road-point to its neighbor in the polyline.
- A one-hot vector indicating whether the road point is a lane-center, a road line, a road-edge that can be collided with, a stop-sign, a crosswalk, a speed-bump, or is of unknown type.

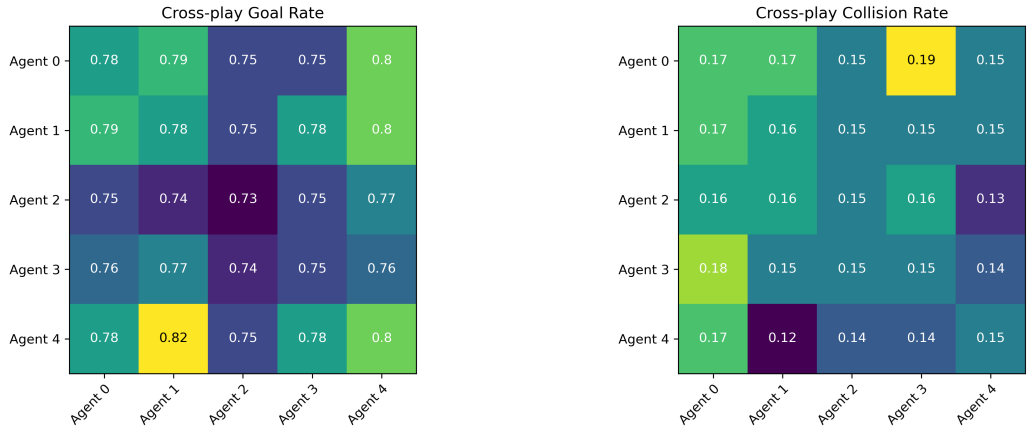


Figure B.1: Goal rate (left) and collision rate (right) of agents when 50% are sampled from one side and 50% from another. The diagonal represents the baseline performance of the seed while element i, j represents seed i playing vs. seed j.

B.2 Zero-shot Coordination

We investigate whether the agents are learning incompatible conventions across seeds (ZSC) by taking the seeds from our best performing agent on the test set (the agent trained on all files) and sampling half the agents from one seed and half from another. We then perform this procedure across all 5 seeds, running each pair on the same set of 100 files. The results of this are summarized in Fig. B.1 and appear to indicate that at the current level of agent performance there does not appear to be a zero-shot coordination issue. However, it is still possible that one might emerge as the agents become more capable and learn arbitrary symmetry-breaking conventions that are not compatible across agents. This argument lines up with the results of Sec. 4.5.3 which shows that the agents fail at high rates in interactive scenes; it may be necessary to get to lower failure rates in such settings for ZSC issues to emerge.

B.3 Ablation of the Targets

The experiments described in the main body of the text provide all the agents with a target position, speed, and heading for their goal. In this section we investigate the effect on some of the metrics of relaxing the speed target requirement. Based on the results from Sec. 4.5.1, we only run experiments using 10000 training files as this appears to be sufficient to close the train-test gap and allows us to minimize the computational requirements of these ablations.

Table B.1 lists a comparison between the original results and the ablation on target speed. Note that since the target requirements are relaxed, the target may become easier to achieve which in turn drives up the goal rate. However, we observe the following features. First, the

collision rate increases sharply relative to APPO and the goal rate decreases markedly. It is possible that the speed target serves as a hint or curriculum that makes solving some of the scenes easier; for example some scenes could be solvable by simply linearly interpolating between the agent speed and the target speed. Since the requirements for reaching the goal are less constrained which allows for a greater range of possible trajectories to reach the target, the ADE increases and the FDE increases significantly. The sharp increase in FDE likely occurs due to the agent being able to reach its goal at higher speed and since the post-goal behavior is out-of-distribution, the agent attempts often unsuccessfully to route itself back to its goal.

Table B.1: Overview of metrics across methods for an 8 second rollout on the test set.

Algorithm	Collision Rate (%)	Goal Rate (%)	ADE (m)	FDE (m)
Expert Playback	4.9	100	0	0
APPO	20.3 ± 0.8	71.7 ± 0.7	3.1 ± 0.2	6.1 ± 0.3
APPO w/o target speed	29.3 ± 1.2	64.9 ± 0.9	4.3 ± 0.3	12.6 ± 0.9
BC	$38.2 \pm .1$	25.3 ± 0.1	5.6 ± 0.1	9.2 ± 0.1

B.4 Vehicle Model

Our vehicles are driven by a kinematic bicycle model [268] which uses the center of gravity as reference point. The dynamics are as follows. Here (x_t, y_t) stands for the coordinate of the vehicle's position at time t, θ_t stands for the vehicle's heading at time t, v_t stands for the vehicle's speed at time t, a stands for the vehicle's acceleration and δ stands for the vehicle's steering angle. L is the distance from the front axle to the rear axle (in this case, just the length of the car) and l_r is the distance from the center of gravity to the rear axle. Here we assume $l_r = 0.5L$.

$$
\begin{aligned}
\dot{v} &= a \\
\bar{v} &= \text{clip}(v_t + 0.5\ \dot{v}\ \Delta t, -v_{\max}, v_{\max}) \\
\beta &= \tan^{-1}\left(\frac{l_r\ \tan(\delta)}{L}\right) \\
&= \tan^{-1}(0.5\ \tan(\delta)) \\
\dot{x} &= \bar{v}\ cos(\theta + \beta) \\
\dot{y} &= \bar{v}\ sin(\theta + \beta) \\
\dot{\theta} &= \frac{\bar{v}\ \cos(\beta)\ \tan(\delta)}{L}
\end{aligned}
$$

We then step the dynamics as follows:

$$
\begin{aligned}
x_{t+1} &= x_t + \dot{x}\ \Delta t \\
y_{t+1} &= y_t + \dot{y}\ \Delta t \\
\theta_{t+1} &= \theta_t + \dot{\theta}\ \Delta t \\
v_{t+1} &= \text{clip}(v_t + \dot{v}\ \Delta t, -v_{\max}, v_{\max})
\end{aligned}
$$

B.5 Calculating Infeasible Goal Statistics

For computing the percentage of vehicles that must cross a road edge to get to their goal, we use the following procedure. We take the vehicles and shrink their width by 0.1 and their length by 0.3. These vehicles are very thin and so do not accidentally collide with a road edge due to small errors in their position. The scaling of the length ensures that the vehicles are not placed in a starting position where they have collided. Using this procedure we calculate the 3% statistic.

B.6 Computing the Simulator Speed

We use two procedures for determining the overall speed of our simulator. The first measures the time to compute observations and step a single agent, which is what is reported as the simulator speed in the paper, and the second presents the time to compute observations and step a group of agents. In the first procedure, we randomly select a set of 2000 files and step through them from the first to the last time-step. At each time-step, we sample a random agent, set the remaining agents to be replayed via expert data, compute the observation for the sampled agent, compute a random action for the sampled agent, and then step the simulator forwards. We repeat this procedure 5 times and report average results and standard deviation of the mean values. Over this dataset and procedure, the single-agent steps-per-second (SPS) is around 3400. This is a slightly modified version of the procedure used to compute FPS in MetaDrive [61] to account for variations in the number of agents available in a scene as well as the lack of rule-based agents in our simulator. With 10 non-agent vehicles we achieve a SPS of 4543, 4049 at 20, and 3497 at 30. The left side of Fig. B.2 shows the scaling with the number of other vehicles in the scene which includes the costs of computing partial observability, collision checking, and stepping the bicycle model.

We also perform a multi-agent variant of this procedure to test the scaling of our simulator with the number of controlled agents. Here we sample observations for all potential controlled agents in a scene and then take a simulator step. Again, we repeat this procedure 5 times, each time sweeping across all scenes in the 2000 files as defined in the prior procedure, and report average results and standard deviation of the mean values over the runs. The right

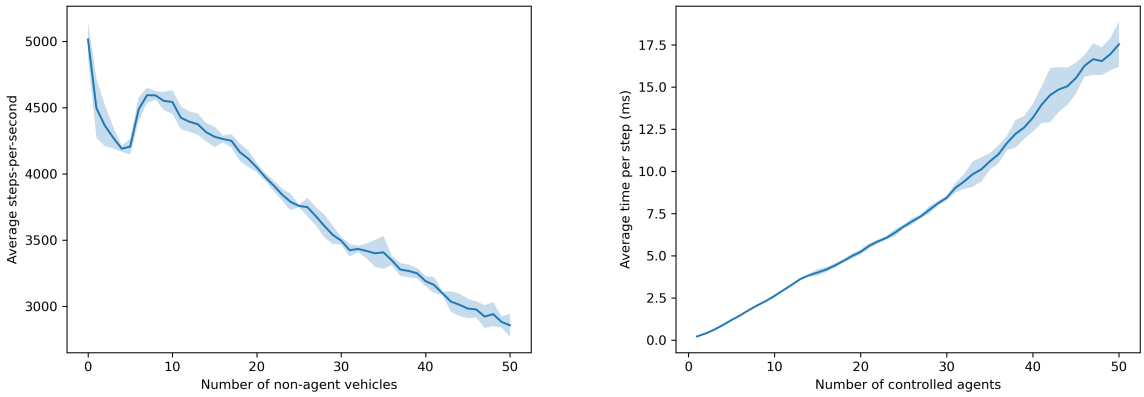


Figure B.2: (Left) **Single-agent**: average steps-per-second as a function of the number of non-agent vehicles in the scene, where each step corresponds to computing observations for a single agent and stepping the simulation. (Right) **Multi-agent**: average time to compute observations and step all the controlled agents in a scene as a function of the number of agents in the scene. Shading corresponds to the standard deviation of the mean. The x-axis is cut at 50 as there are few scenes above this value.

side of Fig. B.2 shows the scaling of the time to perform this procedure as a function of the number of controlled agents in the scene. We observe a seemingly linear scaling with agent number. There are about 8.5 agents on average and we achieve an average per-agent SPS of 452, allowing us to construct about 3800 frames of experience per-second.

Appendix C

Neural Finite Volume

C.1 Finite Volume Methods

Several finite volume-based numerical schemes are studied in this chapter. They include the following common classical first-order schemes:

Godunov [269].

$$\forall i,n \quad \hat{F}^n_{i-1/2} = \begin{cases} \min\limits_{[u^n_{i-1},u^n_i]} f & \text{if } u^n_{i-1} \leq u^n_i \\ \max\limits_{[u^n_i,u^n_{i-1}]} f & \text{if } u^n_{i-1} > u^n_i \end{cases}$$

Lax-Friedrichs [229].

$$\forall i,n \quad \hat{F}^n_{i-1/2} = \frac{1}{2}\left(f(u^n_i) + f(u^n_{i-1})\right) - \frac{1}{2}\frac{\Delta x}{\Delta t} \times |u^n_i - u^n_{i-1}|.$$

Engquist-Osher [270].

$$\forall i,n \quad \hat{F}^n_{i-1/2}(u^n_{i-1}, u^n_i) = \frac{1}{2}\left(f(u^n_i) + f(u^n_{i-1})\right) - \frac{1}{2}\int_{u^n_{i-1}}^{u^n_i} |f'|\,.$$

Additionally, higher-order schemes such as the **Essentially Non-Oscillatory (ENO) method [231]** and the **Weighted Essentially Non-Oscillatory (WENO) method [232]** are considered. The main idea in these methods is that by considering more stencils, one can expect to increase the accuracy of approximation of the solution.

For the ENO scheme, we consider the semi-discrete form of

$$\partial_t u_i = -\frac{1}{\Delta x}\left(\hat{F}_{i+1/2} - \hat{F}_{i-1/2}\right). \tag{C.1}$$

Using the Lax-Friedrichs Splitting technique, we have

$$f(u) = f^+(u) + f^-(u), \quad f^\pm(u) = \frac{1}{2}(f(u) \pm \alpha u), \tag{C.2}$$

where $\alpha = \max|f'(u)|$ is the maximum wave speed. The key point in the ENO scheme is the high-order upwind interpolation of f^+ and f^- based on the smoothest stencils. For instance, for the 2-stencil ENO scheme, the procedure is as follows:

1. Evaluate the smoothness indicators:

$$\delta_- = |f_i^+ - f_{i-1}^+|, \quad \delta_+ = |f_{i+1}^+ - f_i^+|$$

2. Select the stencil that minimizes the smoothness indicator:

   - If $\delta_+ < \delta_-$, choose the stencil $\{f_i^+, f_{i+1}^+\}$.
   - Otherwise, choose the stencil $\{f_{i-1}^+, f_i^+\}$.

3. Perform linear interpolation to compute the numerical flux:

$$\hat{f}^+_{i+\frac{1}{2}} = f_i^+ + \frac{1}{2}\delta^+$$

   where δ^+ is the difference between the selected stencil elements.

A similar approach is applied to compute $\hat{f}^-_{i+\frac{1}{2}}$ using the right-biased stencil.

The final numerical flux at the interface is obtained by combining the positive and negative parts:

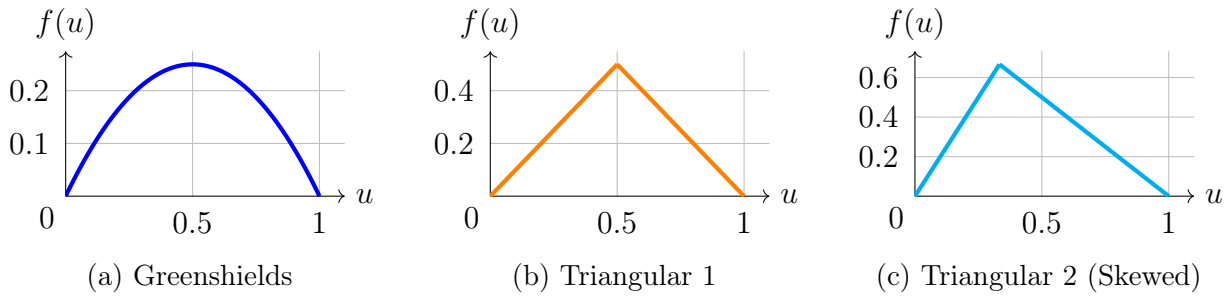
$$\hat{f}_{i+\frac{1}{2}} = \hat{f}^+_{i+\frac{1}{2}} + \hat{f}^-_{i+\frac{1}{2}}$$

In this chapter, we have used a 3-stencil scheme for ENO.

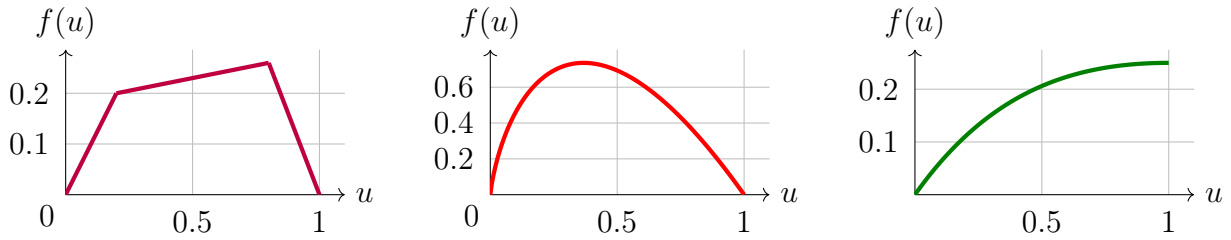
The WENO scheme follows the same idea as ENO by using specific weights in defining $\hat{f}^+_{i+1/2}$, rather than explicit conditions. In this chapter, we use the 5-stencil WENO scheme.

C.2 Variants of LWR

We consider six different LWR PDEs variants, each consisting of a different fundamental diagram, illustrated in Figure C.1. All of the considered flows are concave continuous mappings from $[0, \rho_{\max}]$ to $\mathbb{R}_+$, where $\rho_{\max}$ is the maximum density, with the exception of the Greenberg flow whose domain is $(0, \rho_{\max}]$. The critical density ρ_c denotes the density at which the flow is maximized, i.e. $\rho_c = \arg\max_{\rho \in [0, \rho_{\max}]} f(\rho)$. The following introduces the six flow models we consider in this chapter, each time detailing the flow's parameters, the parameter values we use in Section 8.5 (in parentheses), and the flow's definition. Note that we consider normalized parameter values lying between 0 and 1 for the most part.



(a) Greenshields (b) Triangular 1 (c) Triangular 2 (Skewed)



(d) Trapezoidal (e) Greenberg (f) Underwood

Figure C.1: **Flow models for LWR.** We consider six different variants of the LWR PDE with the flows illustrated here, each mapping road density (veh/m) to traffic flow (veh/s).

Greenshields. Parameters: free-flow speed $v_{\max}$ (1 m/s), maximum density $\rho_{\max}$ (1 veh/m).

$$f(\rho) = v_{\max}\rho\left(1 - \frac{\rho}{\rho_{\max}}\right)$$

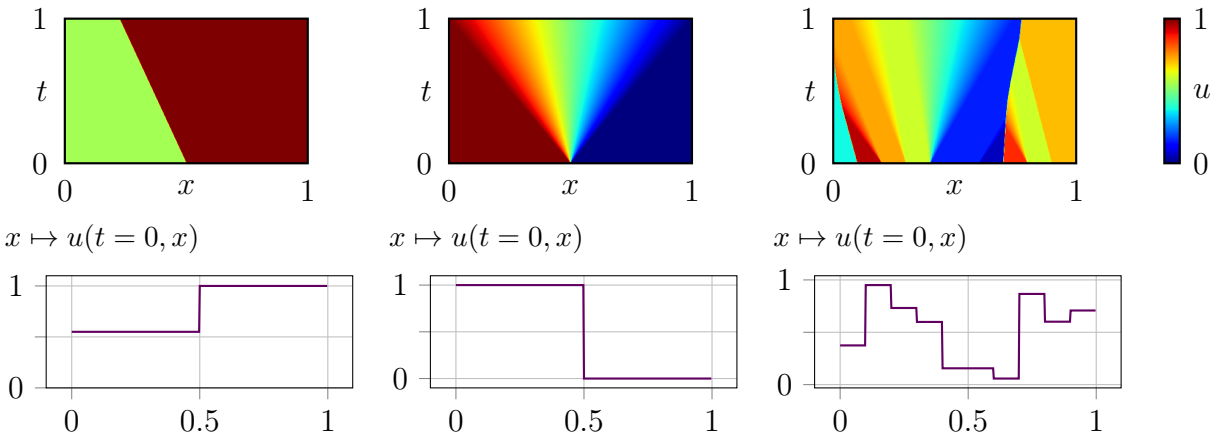
Triangular 1 (symmetrical). Parameters: free-flow speed $v_{\max}$ (1 m/s), critical density ρ_c (0.5 veh/m), maximum density $\rho_{\max}$ (1 veh/m), wave propagation speed (-1 m/s).

$$f(\rho) = \begin{cases} v_{\max}\rho & \text{if } \rho < \rho_c \\ w(\rho - \rho_{\max}) & \text{if } \rho \geq \rho_c \end{cases}$$

Triangular 2 (skewed). A non-symmetric variant of the Triangular flow, with parameters $v_{\max} = 2$ m/s, $\rho_c = 1/3$ veh/m, and $w = -1$ m/s.

Trapezoidal. Parameters: free-flow speed $v_{\max}$ (1 m/s), first density cusp ρ_1 (0.2 veh/m), second density cusp ρ_2 (0.8 veh/m), maximum density $\rho_{\max}$ (1 veh/m), wave propagation speed (-1.5 m/s).

$$f(\rho) = \begin{cases} v_{\max}\rho & \text{if } \rho < \rho_1 \\ (w(\rho_2 - \rho_{\max}) - v_{\max}\rho_1)\dfrac{\rho - \rho_1}{\rho_2 - \rho_1} + v_{\max}\rho_1 & \text{if } \rho_1 \leq \rho \leq \rho_2 \\ w(\rho - \rho_{\max}) & \text{if } \rho > \rho_2 \end{cases}$$



(a) A Shock wave. (b) Rarefaction wave. (c) Eval IC.

Figure C.2: Exact solution for two Riemann problems (left, middle) and one piecewise-constant initial condition (right) from the evaluation set.

Greenberg. Parameters: maximum density $\rho_{\max}$ (1 veh/m), coefficient c_0 (2).

$$f(\rho) = c_0 \rho \log\left(\rho_{\max}/\rho\right)$$

Underwood. Parameters: maximum density $\rho_{\max}$ (1 veh/m), coefficients c_1 (0.25) and c_2 (1).

$$f(\rho) = c_1 \rho \exp\left(1 - c_2 \rho\right)$$

Example solutions of the Greenshields LWR are shown on various initial conditions in Figure C.2.

C.3 More General PDE Models

2D Conservation Laws

A general form of conservation law equations in higher dimension can be written as

$$\begin{cases} \partial_t u + \nabla \cdot f(u) = 0 \\ u(0,x) = u_\circ(x) \end{cases} \tag{C.3}$$

for $(t,x) \in \mathbb{R}_+ \times \mathbb{R}^n$, initial condition $u_\circ$, and $f : \mathbb{R} \to \mathbb{R}^n$. The Riemann solution for such model in higher dimension does no longer exist. Therefore, the unsupervised approach can be very helpful in this case.

Conservation Laws with Discontinuity in the Flux

A more general case of equations of conservation laws will include flux with dependence on the space variable. More precisely,

$$\begin{cases} \partial_t u(t,x) + \partial_x f(\gamma(x), u(t,x)) = 0 \\ u(0,x) = u_\circ(x). \end{cases} \tag{C.4}$$

Such equations are more complex in nature due to the discontinuity of the flux function and consequently lack some analytical properties such as bounded variations. However, in the one-dimensional model, the Riemann solution can still be defined.

ARZ-Type Equations

The conservative form of the Aw-Rascle-Zhang (ARZ) equations [271, 272] can be written in the form of

$$\begin{cases} \partial_t \rho + \partial_x(\rho u) = 0 \\ \partial_t(\rho(u + p(\rho))) + \partial_x(\rho u(u + p(\rho))) = 0 \end{cases} \tag{C.5}$$

where ρ is density, u is the velocity function and $\rho \mapsto p(\rho)$ is know as pressure term. In particular, the first equation represents the mass conservation, while the second equation models the conservation of momentum.

C.4 I-24 Experimental Dataset: Density Extraction, Training, and Evaluation

C.4.1 Density Fields Extraction From I-24 MOTION Dataset

I-24 MOTION is a large-scale traffic monitoring system installed along a section of Interstate 24 near Nashville, Tennessee. It uses a dense network of high-resolution cameras and computer vision algorithms to capture detailed, real-time vehicle trajectories across multiple lanes and miles of highway. The data collection network and resulting trajectory data are illustrated in Figure C.3.

For our experiments, we use the INCEPTION dataset[1] [180] from I-24 MOTION, consisting of ten days of data, each covering the morning rush hour (6:00 AM to 10:00 AM). The dataset for each day comprises 15-20 GB stored as a single JSON file. We first split each file into manageable 1 GB chunks and parse them efficiently using `simdjson` [273], which enables extraction of density fields in approximately 3-5 minutes per 20 GB file.

To construct the density fields, we discretize the spatiotemporal domain into cells of size 0.02 miles (≈32 meters) in space and 0.1 seconds in time, aggregating data across all four

[1] Available at i24motion.org as part of the INCEPTION data release.

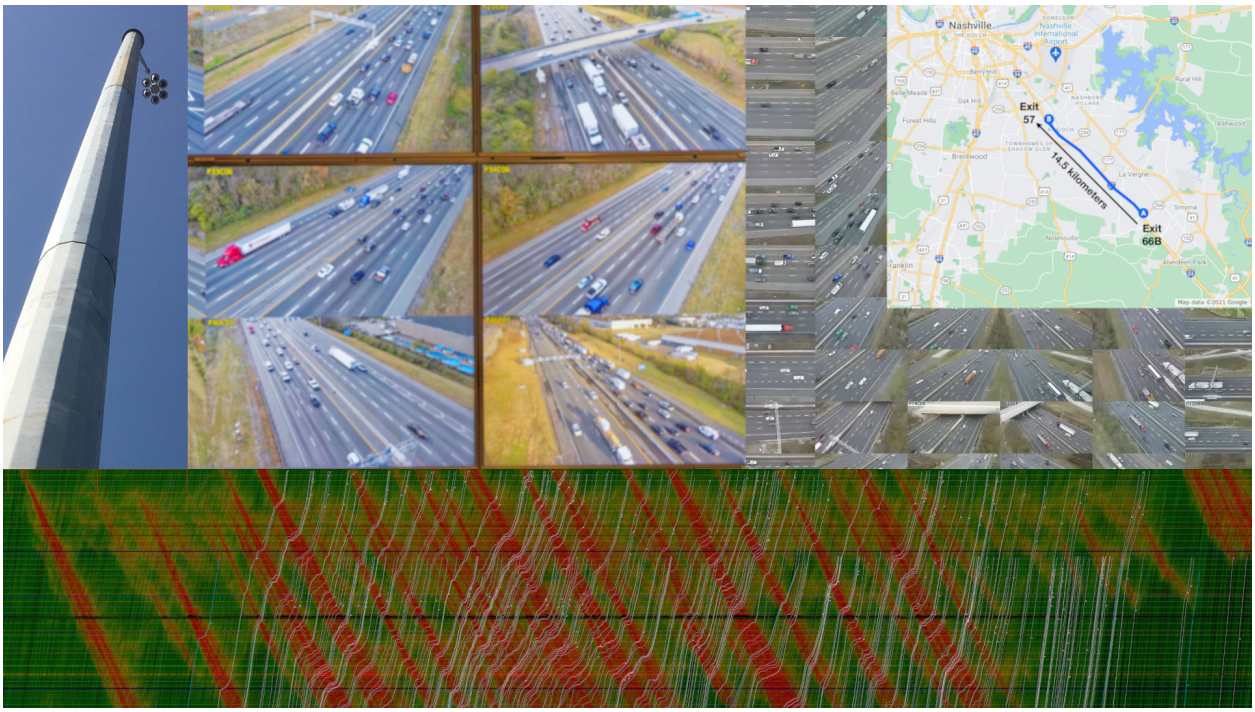
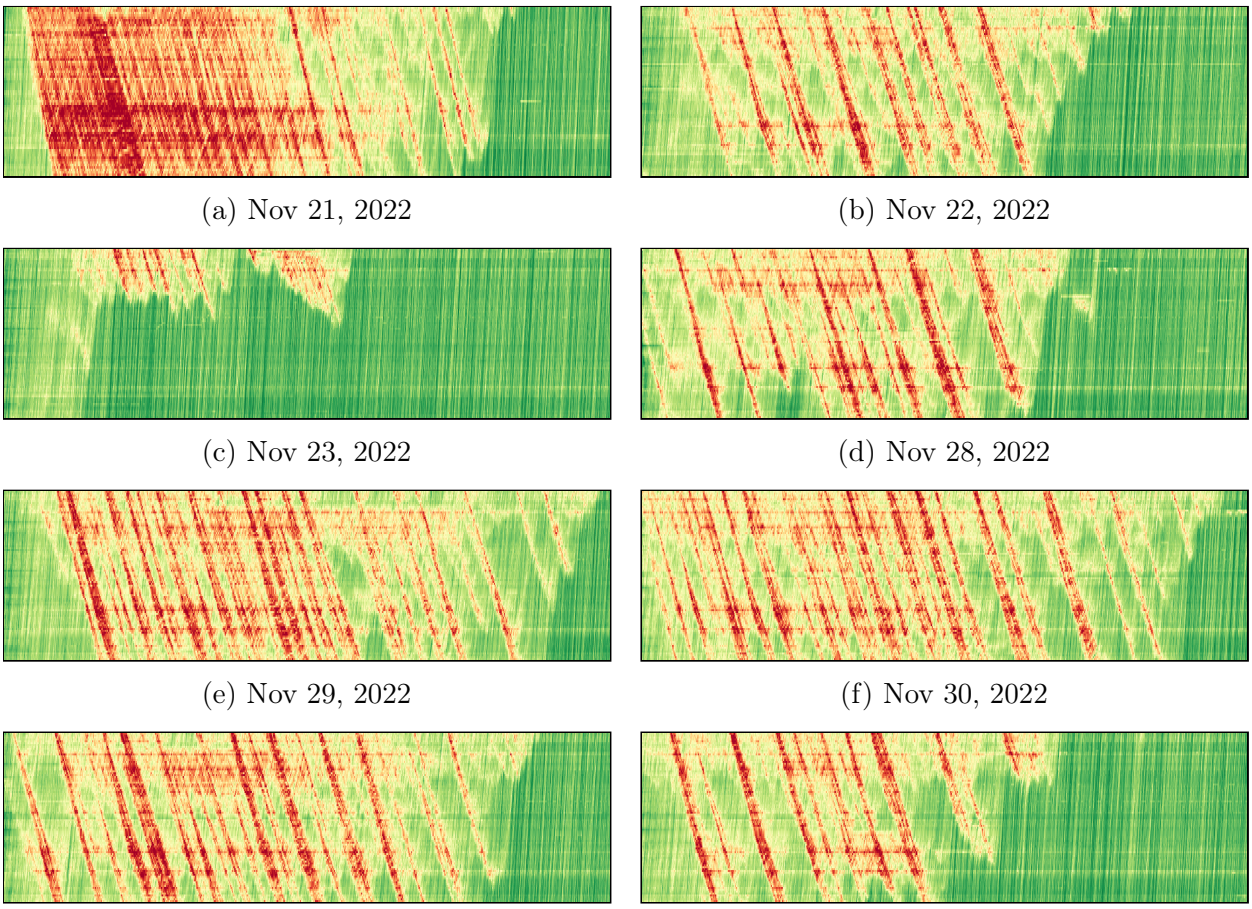


Figure C.3: **I-24 MOTION illustration.** High-definition camera poles are mounted along a portion of I-24 at regular intervals. This generates massive amounts of video data, which is processed through a software stack. The resulting data for a single day is shown in the time-space diagram, displaying thousands of individual vehicle trajectories color-coded by speed (red for low speeds, green for high speeds), illustrating the complexity of the dataset.

lanes. Vehicle counts in each cell are normalized to obtain densities in vehicles per kilometer per lane. To reduce noise, we average over 100 consecutive time steps (i.e., 10 seconds) and over 2 adjacent spatial cells (i.e., 0.04 miles or ≈64 meters). This results in a grid of 100 spatial cells (4 miles / 0.04 miles) and approximately 1440 time steps (4 hours / 10 seconds). We clip the first and last segments of each day to exclude low-density, free-flow regimes with incomplete data, retaining 1300 time steps per day depending on data quality. To avoid extreme outliers, we cap densities at 140 vehicles/km/lane. For all training and evaluation purposes, we then normalize densities so that the maximum density is 1.

Due to occasional sensor failures, such as malfunctioning camera poles or occlusions by bridges, there are seven spatial locations with missing data. We fill these gaps by linear interpolation between the adjacent upstream and downstream cells.

Figure C.4 shows the density fields we extracted from I-24 MOTION data. Higher densities (in red) correspond to stop-and-go waves and congestion, while lower densities (green) correspond to free-flow traffic. The processing code and resulting data are available in our codebase.



(a) Nov 21, 2022 (b) Nov 22, 2022

(c) Nov 23, 2022 (d) Nov 28, 2022

(e) Nov 29, 2022 (f) Nov 30, 2022

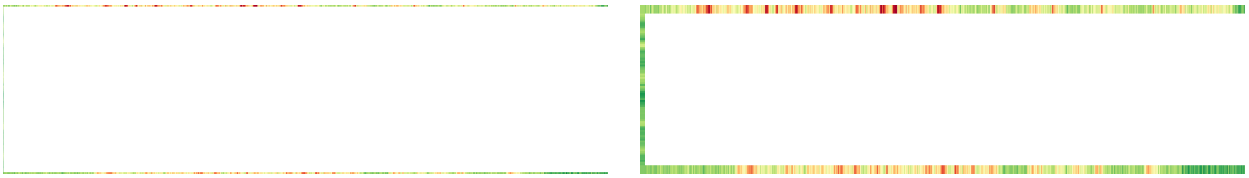
(g) Dec 01, 2022 (h) Dec 02, 2022

Figure C.4: Time-space diagrams of car trajectories extracted from the video, colour-coded by density, for different dates. See Appendix C.4.3 for how to read the heatmaps.

We exclude data from November 24 and 25, 2022, from our analysis, as both days correspond to holiday periods with purely free-flow, low-traffic conditions and no observable stop-and-go waves. These days are therefore not relevant to our study, which focuses on modeling traffic dynamics in the presence of congestion. The remaining days still include sufficient free-flow segments to evaluate model robustness in those regimes. Nevertheless, we include the excluded days in the released dataset for completeness.

C.4.2 Boundary Conditions

For both training and evaluation of NFV on the I-24 dataset, we initialize the model using a single time step of real data and provide one boundary cell at each end of the road, using the corresponding real values. While it is possible to use additional ground truth data to



(a) Nov 29, 2022 boundary conditions for NFV_3^1 (b) Nov 29, 2022 boundary conditions for NFV_{11}^{11}

Figure C.5: **Boundary conditions used by NFV during training and evaluation on the I-24 dataset.** The figures show the input provided to the model: the initial condition at $t = 0$ on the left, and boundary conditions at $x = 0$ (bottom) and $x = x_{\max}$ (top). The model must then predict the interior (i.e., the region shown in Figure C.4) autoregressively: it uses its own output at time t to predict the state at time $t + dt$, without receiving any additional data beyond the fixed boundaries. Note that both figures use the same underlying data; for NFV_{11}^{11}, the boundary values are duplicated to provide the required input padding.

improve accuracy, we deliberately restrict ourselves for two main reasons: (1) to allow fair comparison with the Godunov scheme, which uses a single boundary cell per side, and (2) to reflect realistic deployment scenarios, where boundary densities might only be measured at a few fixed points (e.g., at the road extremities), or predicted using a separate model.

For models that require a wider input stencil (e.g., those larger than NFV_3^1), we pad the boundaries by duplicating the available single-cell values. This ensures that all models, no matter their size-Godunov, NFV3^1, or NFV11^{11}—receive the same amount of boundary information. Figure C.5 illustrates the boundary setup in both cases, showing which values are provided as an input and which are left to be predicted.

Finally, we emphasize that initial and boundary conditions are not included when computing metrics, whether in the training loss or at evaluation.

C.4.3 Reading the Heatmaps

This section provides a brief explanation and intuition for interpreting the heatmaps displaying I-24 MOTION data. The horizontal axis represents time, increasing from left to right, while the vertical axis represents space along the road, increasing from bottom to top. The color encodes traffic density, normalized between 0 and 1, according to the colormap shown in Figure C.6, where green indicates low density traffic (free flow) and red indicates high density traffic (congestion). Unless otherwise specified, only the model predictions are shown, while initial and boundary conditions are omitted for clarity. Stop-and-go waves appear as high-density (red) bands that propagate upstream, i.e., move backward through traffic.

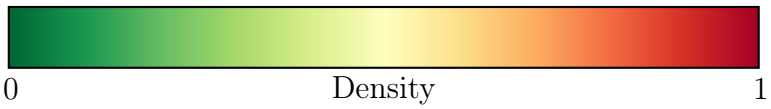


Figure C.6: Colorbar showing density scale for all I-24 data heatmaps.

C.4.4 Predictions

Predictions on the training day from Section 8.6.2 are displayed in Figure C.7. Predictions on evaluation days are displayed in Figure C.8.

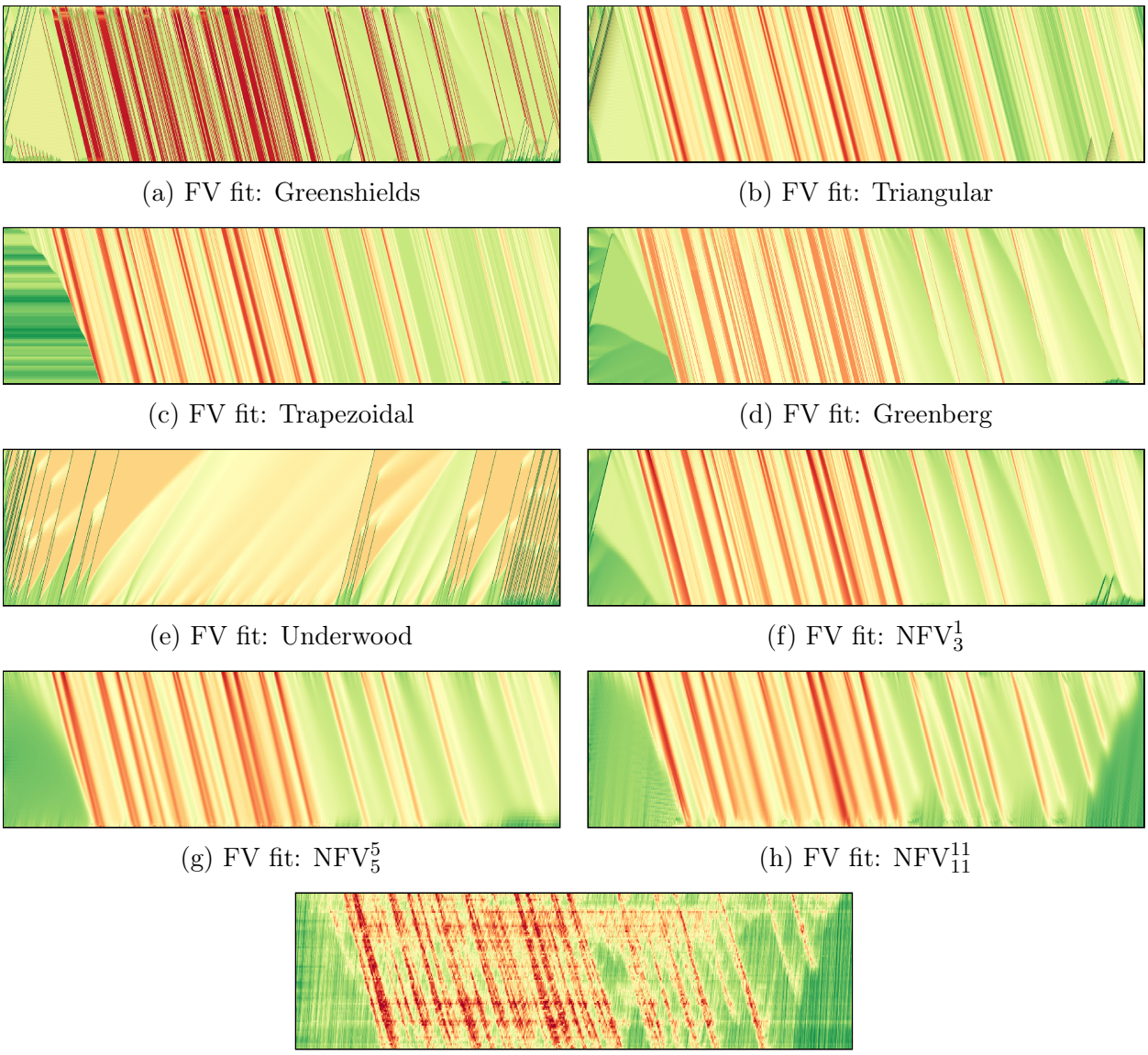
C.5 Experiment Details

Model Architecture. The model is applied locally on each cell to estimate the corresponding numerical flux. It is implemented as a one-dimensional CNN. The first layer uses a kernel of size $a - 1$, followed by five 1D convolutional layers with 15 channels and kernel size 1. Using a CNN enables efficient vectorized computation over all stencils, which is equivalent to sliding a fully connected network along the input but significantly faster. Each time step is represented as a separate input channel, for a total of b input channels. The output is a single channel providing the estimated flux.

Training on Synthetic Data (LWR). For the LWR model, training is performed autoregressively: the model predicts future time steps by feeding its own outputs as inputs. To mitigate error accumulation, the prediction horizon is gradually increased from 10 to 250 steps during training. Initially, most training uses a 10-step horizon, which already yields satisfying performance. Fine-tuning with longer horizons improves stability. The model is trained with the Adam optimizer and an exponentially decaying learning rate, from 10^{-4} to 10^{-8}. Training is done on an RTX 6000 GPU and takes approximately 30 minutes.

Dataset. 400 Riemann problems are randomly sampled for training, with discretization parameters $\Delta t = 5 \cdot 10^{-3}$ and $\Delta x = 10^{-2}$, with 100 space cells and up to 250 time steps. Training details can be found in Appendix C.5. A finer grid is used to compute the 100 test cases for evaluation using the Lax-Hopf algorithm: $\Delta t = 10^{-4}$ and $\Delta x = 10^{-3}$ with 200 cells and $1,000$ time steps.

Training on Experimental Data. All models and fitted finite volume schemes are trained on the first hour of data from November 29, 2022 (arbitrarily selected), and evaluated on the full morning period (nearly four hours) and the remaining days of data. To ensure fairness and reflect practical deployment constraints, each model receives only a single boundary cell on each side, as described in Appendix C.4.2, even though larger models could benefit



(a) FV fit: Greenshields

(b) FV fit: Triangular

(c) FV fit: Trapezoidal

(d) FV fit: Greenberg

(e) FV fit: Underwood

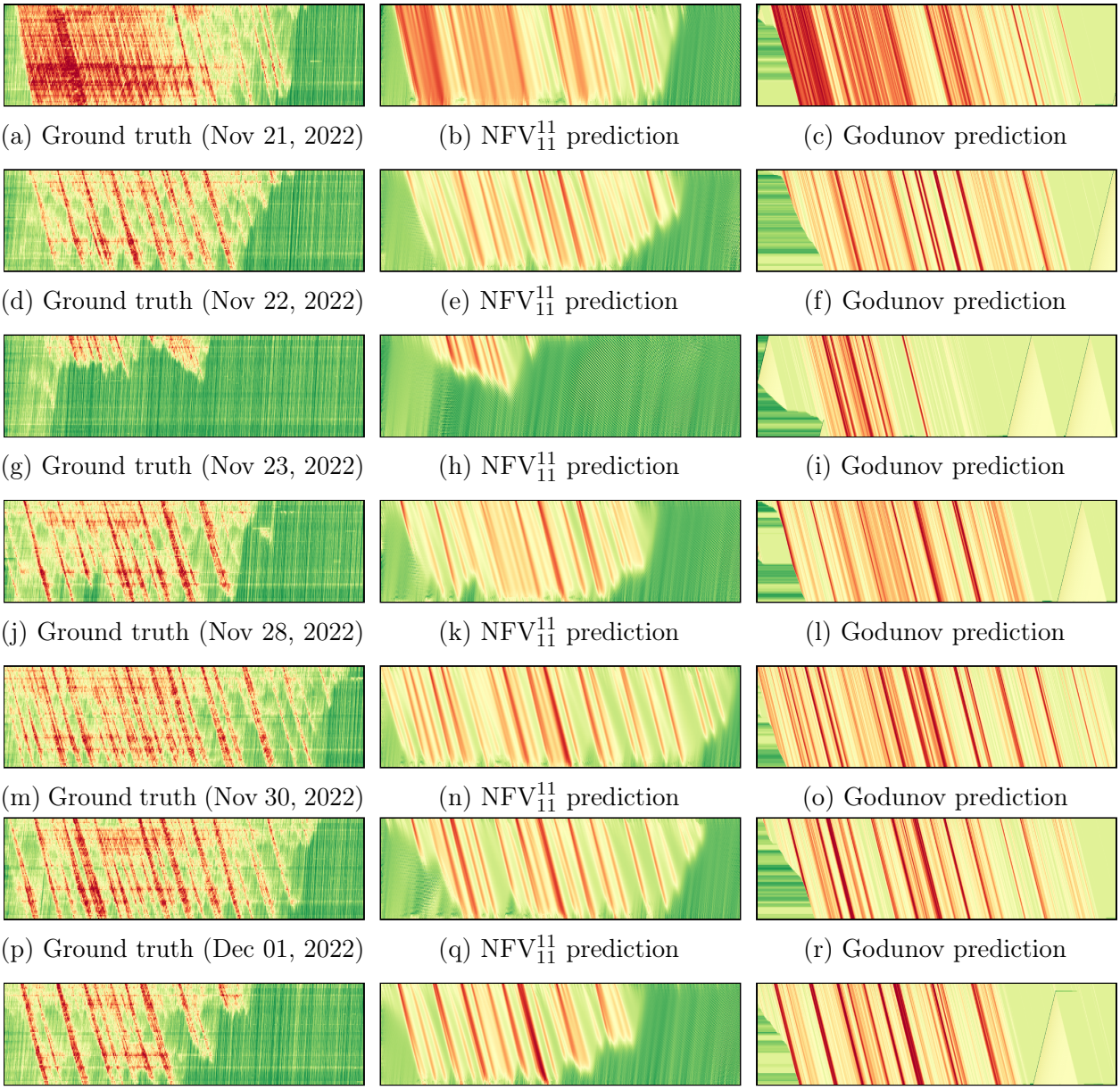
(f) FV fit: NFV_3^1

(g) FV fit: NFV_5^5

(h) FV fit: NFV_{11}^{11}

(i) Ground truth

Figure C.7: **Predictions of FV methods and trained NFV.** Corresponding metrics are reported in Table 8.3. Among the FV methods, only the Triangular, Trapezoidal, and Greenberg flows provide a reasonable fit to the I-24 MOTION data. In contrast, NFV models show increasing predictive accuracy with model complexity. For example, NFV_{11}^{11} captures significantly more stop-and-go waves (in red) than NFV_5^5 or NFV_3^1, as well as fast low-density waves (in green), enabling it to correctly predict the early dissipation of the final two waves. However, it exhibits oscillations toward the end of the prediction window, likely due to limited generalization caused by the scarcity of low-density (dark green) patterns in the training data; nevertheless, the primary objective when modeling experimental data is to accurately capture the evolution of congestion waves, whereas free-flow traffic is of lesser interest. All models were trained on only the first 25% of the ground truth sequence, and the predictions are generated fully autoregressively. See Appendix C.4.3 for how to read the heatmaps.



(a) Ground truth (Nov 21, 2022) (b) NFV_{11}^{11} prediction (c) Godunov prediction

(d) Ground truth (Nov 22, 2022) (e) NFV_{11}^{11} prediction (f) Godunov prediction

(g) Ground truth (Nov 23, 2022) (h) NFV_{11}^{11} prediction (i) Godunov prediction

(j) Ground truth (Nov 28, 2022) (k) NFV_{11}^{11} prediction (l) Godunov prediction

(m) Ground truth (Nov 30, 2022) (n) NFV_{11}^{11} prediction (o) Godunov prediction

(p) Ground truth (Dec 01, 2022) (q) NFV_{11}^{11} prediction (r) Godunov prediction

(s) Ground truth (Dec 02, 2022) (t) NFV_{11}^{11} prediction (u) Godunov prediction

Figure C.8: **Predictions of best FV fit and trained NFV_{11}^{11}.** Godunov is derived by fitting a flow function on the prediction and comparing it against the ground truth; we keep the fitted Trapezoidal flow as it performed best (see Figure C.7 and Table 8.3). Both the Godunov fit and the NFV_{11}^{11} training are realized using the same data, namely the first 1 hour of Nov 29, 2022 data. This means that the remainder of the data on that day, as well as the prediction on all subsequent days (shown in this figure) are generalization on data that was never seen before by either models. See Appendix C.4.3 for how to read the heatmaps.

from additional context. Each NFV model consists of 6 hidden layers of width 15, totaling $1105 + 16 \cdot ((a-1) \cdot b + 1)$ parameters for NFV_a^b. Training takes 15 to 30 minutes on an RTX 4080 GPU. As with synthetic data, the prediction horizon increases progressively during training, from 10 to 100 steps, and the learning rate decays from 10^{-3} to 10^{-4} over 3000-5000 epochs depending on model size, until convergence.